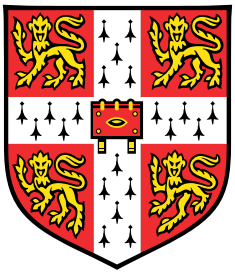
CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Apollo

Symmetric Circuits and Model-Theoretic Logics



Gregory Wilsenach

University of Cambridge

This thesis is submitted for the degree of
Doctor of Philosophy

Peterhouse

February 2019

To my parents, grandparents, and brother.

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee.

Gregory Wilsenach
February 2019

Acknowledgements

I would first like to thank my supervisor, Anuj Dawar. Anuj has been unendingly supportive throughout my time in Cambridge, and has been both an incredible teacher and a wonderful mentor. I am so lucky to have benefited from his seemingly unlimited patience and encouragement. I continue to be in awe of his keen insight and ability to clearly understand and explain complex ideas. I am exceedingly grateful to have had Anuj as a supervisor.

I should also like to thank many people in my research group. I would like to mention: Arno Pauly, Pengming Wang, Jannis Bulian, Danny Vagnozzi, Adam ó Conghaile, and Abhisekh Sankaran. I would especially like to mention my long-suffering office mate, Mathew. I have really enjoyed my meetings with Wied Pakusa, who is always both kind and eager to teach. I am also very grateful to have met Joanna Ochremiak, Nathanael Fijalkow, Anupam Das, and many others in the logic community.

I would like to thank the Gates Cambridge Trust for their financial support. I would also like to thank the Gates Cambridge community more broadly. I have enjoyed getting to know this wonderful group of people, and I hope to keep in contact in the years to come. I would especially like to mention my friends from the Gates Room. So many late nights!

I would like to thank my college, Peterhouse, for their financial support and general kindness. The MCR has been wonderful and I have thoroughly enjoyed my time.

I am so grateful for the incredible friends I have been fortunate enough to have here. I would especially like to mention Amandla, Arif, Bhasi, Jon, Taskeen, and Tariq.

To my parents, grandparents, and brother, of course, without you all I certainly wouldn't be here. I owe everything to you.

Lastly, to Aulia, my partner, whose been so supportive and has been making trips to Cambridge for what seems like forever in order to give me time to work. You have been so patient and utterly wonderful. I will always be so grateful.

Abstract

The question of whether there is a logic that characterises polynomial-time is arguably the most important open question in finite model theory. The study of extensions of fixed-point logic are of central importance to this question. It was shown by Anderson and Dawar that fixed-point logic with counting (FPC) has the same expressive power as uniform families of symmetric circuits over a basis with threshold functions.

In this thesis we prove a far-reaching generalisation of their result and establish an analogous circuit characterisation for each from a broad range of extensions of fixed-point logic.

In order to do so we fist develop a very general framework for defining and studying extensions of fixed-point logics, which we call *generalised operators*. These operators generalise Lindström quantifiers as well as the counting and rank operators used to define FPC and fixed-point logic with rank (FPR).

We also show that in order to define a symmetric circuit model that goes beyond FPC we need to consider circuits with gates that are allowed to compute non-symmetric functions. In order to do so we develop a far more general framework for studying circuits. We also show that key notions, such as the notion of a symmetric circuit, can be analogously defined in this more general framework. The characterisation of FPC in terms of symmetric circuits, and the treatment of circuits generally, relies heavily on the assumption that the gates in the circuit compute symmetric functions. We develop a broad range of new techniques and approaches in order to study these more general symmetric circuit models.

As a corollary of our main result we establish a circuit characterisation of FPR. We also show that the question of whether there is a logic that characterises polynomial-time can be understood as a question about the symmetry property of circuits. We lastly propose a number of new approaches that might exploit this new-found connection between circuit complexity and descriptive complexity.

Table of contents

Chapter 1

Introduction

The P vs NP conjecture is arguably one of the deepest open problems in mathematics and theoretical computer science. The class P can be informally understood to consist of all those problems that are computationally "easy" to solve and the class NP all those problems with the property that any potential solution is "easy" to verify for correctness. The conjecture is that there exists a problem with the property that any potential solution is easy to verify but the problem itself is hard to solve, i.e. is there problem in NP that is not in P?

The formal definitions of P and NP are given in terms of time-bounded Turing machines. The class P consists of all those decision problems decidable by a polynomial-time bounded deterministic Turing machine and the class NP consists of those problems with the property that a certificate can be *verified* by a polynomial-time bounded deterministic Turing machine. In order to prove this conjecture we would need to establish some useful characterisation of polynomial-time decidability. The difficulty is that the given definition in terms of machine models is hard to analyse and offers little insight.

There are numerous approaches to complexity theory that try to avoid machine models altogether. The field of *descriptive complexity* studies the expressive power of logics and aims to classify complexity classes, or fragments of them, in terms of these logics. This logical approach offers us some insight into what sort of "abstract machinery" is needed to solve those problems in a class (e.g. do we need recursion, counting, etc.). The foundational result in this field was proved by Fagin who established a characterisation of NP in terms of existential second-order logic [18]. More formally, he showed that a class of structures is definable in existential second-order logic if, and only if, the corresponding decision problem is in NP. Fagin's theorem has led to similar logical characterisations for a variety of other complexity classes, such as coNP, PH, PSPACE, and EXP [32]. The question of whether there is a similar logical characterisation for polynomial-time is arguably the single most important open problem in descriptive complexity.

1.1 A Logic for P?

The question of whether there is a logic for polynomial-time was originally posed in database theory in a slightly different form. In response to the observation that many standard query languages could not express every polynomial-time decidable query, Chandra and Harel [10] asked whether there exists a recursive enumeration of those classes of structures decidable in polynomial-time. This was later reformulated by Gurevich [26] in descriptive complexity as the question of whether there is a logic that decides exactly those classes of structures that can be decided in polynomial time (with some computational caveats). In this case we say that the logic *captures* polynomial-time.

The search for a logic that captures P has focused on extensions of first-order logic (FO). It can be shown that each class of structures definable in FO is decidable in polynomial-time. However, it is also easy to show that FO cannot decide the class of structures with an even size domain or the class of connected graphs, both of which are obviously decidable in polynomial-time. Fixed-point logic (FP) extends FO with a mechanism for recursively defining a predicate. This allows FP to define, for example, the transitive closure of a relation. It follows that FP can decide the class of connected graphs and so the expressive power of FP properly extends FO. Moreover, it was shown by Immerman and Vardi [30, 42] that FP captures polynomial-time on the class of ordered structures. In other words if a class of structures $\mathcal{C}$ is polynomial-time decidable and is defined over a vocabulary including a binary symbol $\leq$ that is always interpreted as a linear order then $\mathcal{C}$ is definable in FP. However, despite these strengths, FP cannot define the class of even size structures.

Immerman [31, 30] proposed extending FP in order to add mechanisms for reasoning about quantities and for defining the cardiniality of a definable set. This extension is called fixed-point logic with counting (FPC) and it has become one of the most well-studied logics in descriptive complexity. It has been shown that FPC captures polynomial-time over a number of natural graph classes (e.g. [23, 25]) and, most generally, it was shown by Grohe that FPC captures polynomial-time over any class of graphs defined by excluded minors [24]. In fact, it was shown by Hella et al. [28] that FPC captures polynomial-time on "almost all structures" in some precise sense. The centrality of FPC has led to it being considered the logic of reference in the search for a logic for P [12].

Despite this, it was shown by Cai, Fürer, and Immerman [9] that FPC does not capture polynomial-time. Their proof involved constructing a class of graphs for which the isomorphism problem is polynomial-time decidable but not expressible in FPC. This construction has become standard and has spawned numerous variants that have been used to establish other inexpressibility results (e.g. [22, 29]). It was shown by Atserias et al. [5] that in each of these cases the problem can be reduced to a question about the solubility of a system of linear equations over a finite field.

This observation led Dawar et al. [13] to introduce fixed-point logic with rank. This logic is defined by extending FP with a family of rank operators each of which define the rank of a definable matrix over a particular finite field. It was shown by Grädel and Pakusa [22] that this logic does not capture polynomial-time, and they proposed an alternative formulation with a single rank operator that takes a prime as a parameter. We call this logic FPR. Since FPR can express the rank of a matrix, and hence the solubility of systems of linear equations over finite fields, the usual variants of Cai, Fürer, and Immerman, cannot be used to separate FPR from P. It remains an open question whether FPR captures all of polynomial-time and, given the inapplicability of standard graph constructions, new approaches are needed.

1.2 Symmetric Circuits and Fixed-Point Logic

The field of *circuit complexity* similarly avoids machine models and attempts to address complexity-theoretic questions by framing them in terms of Boolean circuits. A Boolean circuit is a directed acyclic graph with a designated set of input gates each labelled by a variable, with all other gates labelled by the elements from some basis, and with a designated output gate. The basis is often taken to consist of the familiar Boolean functions AND, OR and NOT. We call this basis the *standard basis*. A circuit with n input gates defines a function $f : \{0,1\}^n \to \{0,1\}$ for each input $\vec{x} \in \{0,1\}^n$ by setting the input gates in accord with $\vec{x}$ and then recursively evaluating the gates of the circuit.

It can be shown that the behaviour of any algorithm on inputs of a fixed size can be described by a Boolean circuit. The behaviour of an algorithm on arbitrary size inputs can thus be characterised by a family of circuits $(C_n)_{n\in\mathbb{N}}$, where each C_n describes the computation for inputs of size n. This description of an algorithm exposes the combinatorial structure of the computation being performed, and offers means of proving impossibility results. This approach has yielded numerous lower bounds for circuit families of bounded depth [19, 40] and restricted monotone gates [2].

We can instead consider circuits that take structures as input, rather than strings. In this case the input gates of the circuit are labelled by the potential elements of the relations of a structure. For example, a circuit that takes as input a directed graph of size n has n^2 input gates each labelled by some $(i,j) \in [n]^2$. For a particular graph the input gate labelled by (i,j) is set to one if, and only if, there is an edge between the vertices i and j in the graph. We say a circuit is *invariant* if the function it computes in invariant under isomorphism. It can be shown that a class of structures is decidable in polynomial-time if, and only if, it is definable by a polynomially-uniform family of invariant circuits. We recall that a family of circuits $(C_n)_{n\in\mathbb{N}}$ is *polynomially-uniform* (or P-*uniform*) if the function $n \mapsto C_n$ is computable in time polynomial in n.

We say a circuit C is *symmetric* if any permutation on the n elements extends to an automorphism of C. This is intended to formalise the notion of a circuit that is not just

invariant, but whose structure ensures isomorphism-invariance at each step. The restriction to symmetric circuits arises naturally in the study of logics and has appeared previously under the names "regular circuits" in the work of [17] and "explicitly symmetric circuits" in the work of Otto [39]. In particular, there is a natural way of translating any formula in FO or FP into an equivalent polynomially-uniform family of symmetric circuits[32]. Similarly, the formulas in FPC can be translated to polynomially-uniform families of symmetric circuits over the extension of the standard basis with threshold functions [32]. In this way we can think of the formulas in these logic as defining inherently symmetric algorithms.

Anderson and Dawar [3] showed that, in fact, these families of symmetric circuits are exactly characterised by these logics. In particular, they showed that a class of structures is definable by a polynomially-uniform family of symmetric circuits defined over the standard basis with threshold functions if, and only if, it is definable in FPC. They also establish a similar characterisation for the extension of FP with a number sort in terms of symmetric circuits defined over the standard basis. These results establish an interesting relationship between two well-known fields in complexity theory and allow us to understand inexpressibility results for logics as lower-bounds for symmetric circuits.

1.3 The Contributions and Structure of this Thesis

The computational power of polynomially-uniform families of invariant circuits is the same no matter if we consider circuits defined over the standard basis or over the basis with threshold functions. In contrast, it follows from the characterisations given by Anderson and Dawar that the choice of basis does affect the expressive power of families of symmetric circuits. In this thesis we study a generalisation of the usual notion of a symmetric circuit that allows for circuits to be defined over a much broader range of bases. We establish a far-reaching generalisation of the characterisation of FPC and show that any from a large class of extensions of fixed-point logic can be characterised by families of symmetric circuits over an appropriate basis. Importantly, previous circuit characterisations (including those in [3] and [39]) rely fundamentally on the assumption that the circuits are defined over a basis of symmetric Boolean functions. This assumption is not particular to them, but is pervasive in circuit complexity (see [43, 33]). In order to prove this characterisation in a more general setting we first need to develop an entirely novel framework for circuits. We now give a brief overview of the approach taken in this thesis.

We first develop a general framework for defining and studying extensions of model-theoretic logics. We note that while there is a general notion of a quantifier by Lindström that generalises the usual existential and universal quantifiers, there is no corresponding notion of operators that generalises the counting or rank operators. In Chapter 3 we develop a framework of *generalised operators* and show that both counting and rank operators can be defined in this framework. In Chapter 4 we discuss symmetric circuits. We begin by

discussing a crucial, and often unstated, assumption imposed by the structure of a circuit. In particular, since a circuit is a directed acyclic graph, there is no structure imposed on the inputs of any internal gate. It follows that, in order to ensure unambiguous evaluation of any gate, the basis must be comprised of *symmetric functions*, i.e. functions invariant under any permutation of the input string. We also show in Chapter 4 that any polynomially-uniform family of symmetric circuits over a basis of symmetric functions can only express queries in FPC.

As such, in order to extend these circuit characterisations and go beyond FPC, we need to be able to define symmetric circuits over bases that include non-symmetric functions. This requires developing a general framework of *structured functions*. These functions, while not necessarily symmetric, can be associated with an appropriate weaker notion of symmetry. We show that any set of generalised operators defines a corresponding basis of structured functions. We generalise the notion of a Boolean circuit in order to allow for bases of structured functions. We correspondingly generalise the important symmetry-related notions.

While we do generalise many of the important results proved by Anderson and Dawar [3], the proof methods they use crucially rely on the symmetry assumption on the basis functions. In Chapter 4 we discuss this point and note that many algorithmic properties of symmetric circuits do not hold in our more general setting (unless the graph-isomorphism problem is in polynomial-time). We introduce the notion of a transparent circuit and, in order to ensure these algorithmic properties hold, we restrict our attention to transparent circuits.

We earlier discussed the fact that there is a standard approach for translating formulas from fixed-point logic to polynomially-uniform families of symmetric circuits. However, this translation does not, in general, produce *transparent* symmetric circuits. In Chapter 5 we present a new, more technically involved, translation that takes formulas in an extension of fixed-point logic to polynomially-uniform families of *transparent* symmetric circuits.

The translation in [3] from families of symmetric circuits to FPC relies on the *support theorem*. This result establishes that for any polynomial-size family of symmetric circuits there exists $k \in \mathbb{N}$ such that for every gate g in the symmetric circuit C_n (the circuit in the family that takes n-size input structures) there exists a set $X \subseteq [n]$ such that $|X| \leq k$ and any permutation that fixes X pointwise extends to a permutation that fixes g. In order to establish a translation from our more general symmetric circuits to extensions of fixed-point logic we similarly need a support theorem. However, Anderson and Dawar's proof uses in an essential way the fact that the functions computed at each gate in the circuit are symmetric. In Chapter 6 we develop new machinery in order to overcome this difficulty and prove the support theorem in our more general setting. We also define a group action on the labels introduced in our more general circuit model and show that the support theorem can be extended so as to apply to these labels as well.

In Chapter 7 we establish that a number of circuit properties (e.g. the orbits, the action of an automorphism, etc.) are polynomial-time decidable for transparent circuits. We

prove that many of these properties are at least as hard to decide over arbitrary circuits as the graph-isomorphism problem. We use these observations to motivate the restriction to transparent circuits.

Finally, with all of the tools in place, in Chapter 8 we formally prove the translation from polynomially-uniform families of transparent symmetric circuits to extensions of fixed-point logic, and hence complete the characterisation. The approach taken by Anderson and Dawar [3] makes crucial use of the symmetry of the functions computed by the gates in the circuit. To be precise, they use the fact that to evaluate a gate g it suffices to count the number of inputs to g that evaluate to true and the fact that for each gate g there is a bijection between the orbit of a g and the set of assignments from the universe of the input structure to the support of g. In our more general context counting the number of true inputs does not suffice to determine the evaluation of a gate. We develop a novel approach that allows us to recursively define the structure on the inputs of each gate, which can then be used to recursively evaluate the gates in the circuit. This new approach requires developing a complete new set of technical tools.

In Chapter 9 we state and prove the main result formally and discuss a few specific cases and corollaries.

1.4 Previously Published Work

The mathematical content presented in many of the chapters in this thesis is based on work done in collaboration with Anuj Dawar. The work done with Anuj Dawar concerned the special case of transparent symmetric circuits defined over an extension of the standard basis with rank threshold gates, and established a characterisation FPR in terms of polynomially-uniform families of these circuits. The main result of this thesis is far more general and establishes circuit characterisations for extensions of fixed-point logic by a very broad range of operators. However, there are many theorems and ideas that carry over to our more general setting and are presented in this thesis. The work done with Anuj Dawar was presented at the 27th annual EACSL conference in Computer Science Logic [16]. A more complete version of this paper is available as a preprint [15]. We briefly describe the relationship between these papers and the work in this thesis.

The work in Chapter 3 is not contained in any of these papers. With the exception of the discussion of generalised operators and bases, most of Chapter 4 is based on work presented in [15]. The translation from formulas to circuits in Chapter 5 is structurally similar to the corresponding translation in [15], but the approach presented here differs significantly. It also establishes a translation for a broad range of extensions of fixed-point logic, not just FPR. The proof of the support theorem in Chapter 6 is similar to the proof of the support theorem in [15], but with some changes in notation and in some of the preceding lemmas. The results in Chapter 7 are all closely based on work presented in [15]. The translation from

circuits to formulas in Chapter 8 uses similar ideas as the corresponding translation in [15], but the translation presented here is far more general and is suited to a much broader range of circuits.

Chapter 2

Preliminaries

In this chapter we provide a brief introduction to the relevant background material from mathematical logic, complexity theory, and algebra. We also define terms and introduce notation that will be used throughout this thesis. These definitions are, for the most part, considered standard. The more expert reader may prefer to refer back to this chapter as needed. We refer the reader to the following sources [31, 20, 29] for a more detailed discussion of many of the topics briefly introduced here.

2.1 Basic Notation

Let $\mathbb{N} = \{1, 2, \ldots\}$ and $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ be the positive and non-negative integers, respectively. For each $a, b \in \mathbb{N}_0$ let $[a, b] := \{x \in \mathbb{N}_0 : a \leq x \leq b\}$. For each $n \in \mathbb{N}_0$ let $[n] = [1, n]$ and let $[n]_0 = [0, n]$.

Let X be a set and $k \in \mathbb{N}$. A k-tuple in X is a function $\vec{v} : [k] \to X$. We denote $\vec{v}$ by $\vec{v} = (v_1, \ldots, v_k)$ where for each $i \in [k]$, $v_i = \vec{v}(i)$. We abuse notation and write $x \in \vec{v}$ if, and only if, there exists $i \in [k]$ such that $v(i) = x$. We let $|\vec{v}| = |\mathbf{Dom}(\vec{v})|$ and call $|\vec{v}|$ the length of $\vec{v}$. If $\vec{v} = (v_1, \ldots, v_n)$ and $\vec{u} = (u_1, \ldots, u_m)$ are tuples we write $\vec{v} \cup \vec{u}$ to denote the set $\{v_1, \ldots, v_n, u_1, \ldots, u_m\}$.

Let X and Y be sets. If $X \subseteq Y$ we write id_X to denote the *identity function* $\mathrm{id}_X : Y \to \{0, 1\}$ defined for all $x \in Y$ such that $\mathrm{id}_X(x) = 1$ if, and only if, $x \in X$. We say that a relation $R \subseteq X \times Y$ is *trivial* if $R = X \times Y$. We write X^Y to denote the set of all functions from Y to X and $X^{\underline{Y}}$ to denote the set of injections in X^Y. For $k \in \mathbb{N}_0$ we write X^k to denote $X^{[k]}$ and $X^{\underline{k}}$ to denote the set of injections from $[k]$ to X. If a relation $R \subseteq X^k$ for some $k \in \mathbb{N}_0$ we say that R has *arity* k and say that R is a k-ary relation. If R is a 0-ary relation we say that R is a *nullary* relation. We similarly say that a function $f : X^k \to X$ has *arity* k and call f a k-ary function. If f is a 0-ary function we call f a *nullary* function. If $f : X \to Y$ is a function and $\approx$ is an equivalence relation on Y we write $f/_\approx$ to denote the

function $f/_{\approx} : X \to Y/_{\approx}$ defined such that $f/_{\approx}(x) = [f(x)]$ for all $x \in X$. We write $\mathcal{P}(X)$ to denote the powerset of X.

2.2 Group Theory

We assume a some basic knowledge of group theory. For an introduction to group theory please see [41]. We now review some group-theoretic notation that will be used in this thesis.

Let X be a set. We write $\mathbf{Sym}_X$ to denote the symmetric group on X. For $n \in \mathbb{N}$ we write $\mathbf{Sym}_n$ to abbreviate $\mathbf{Sym}_{[n]}$. If G and H are groups we write $H \leq G$ to denote that H is a subgroup of G.

Let G be a group. Let X be a set on which a group action of G is defined. For $S \subseteq X$ let $\mathbf{Stab}_G(S) := \{\sigma \in G : \forall x \in S, \sigma x = x\}$ be the (pointwise) *stabiliser* of S with respect to G. We can lift the action of G on X to an action on the powerset of X by letting $\sigma S := \{\sigma x : x \in S\}$ for each $\sigma \in G$ and $S \subseteq X$. For $S \subseteq X$ let $\mathbf{SetStab}_G(S) := \mathbf{Stab}_G(\{S\}) = \{\sigma \in G : \sigma S = S\}$ be the *set-wise stabiliser* of S with respect to G. Let $\mathbf{Orb}_G(S) := \{\sigma S : \sigma \in G\}$ be the *orbit* of S with respect to G. If S is a singleton we omit the set braces, e.g. we write $\mathbf{Stab}_G(x)$ for $\mathbf{Stab}_G(\{x\})$. If $G = \mathbf{Sym}_n$ for some $n \in \mathbb{N}$ we replace the subscript with the number n, e.g. we write $\mathbf{Orb}_n(x)$ for $\mathbf{Orb}_{\mathbf{Sym}_n}(x)$, and if the group G is obvious from context we omit the subscript entirely.

2.3 Logic

In this section we introduce some of basic concepts and notion from mathematics generally and finite model theory in particular.

2.3.1 Vocabularies

A *vocabulary* τ is a set of relation and function symbols. Each relation or function symbol T in τ is associated with number $\text{arty}(T) \in \mathbb{N}_0$ called the *arity* of the symbol. We write $\tau = (\boldsymbol{R}, \boldsymbol{F})$ where $\boldsymbol{R}$ is a set of relation symbols and $\boldsymbol{F}$ is a set of function symbols to denote the vocabulary consisting of those symbols in $\boldsymbol{R}$ and $\boldsymbol{F}$. We usually denote a relation symbol by R and a function symbol by F. We denote the arity of a relation symbol R by r_R and the arity of a function symbol F by f_F. If a relation or function symbol has arity 0 we say it is a *nullary* symbol. We call a nullary function symbol a *constant symbol*. We often use lower case letters to denote constant symbols. We say that a vocabulary τ is *relational* if every symbol in τ is a relation symbol. Unless otherwise stated we assume all vocabularies are finite. A *many-sorted vocabulary* τ is a sequence $(\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ where $\boldsymbol{R}$ is a set of relation symbols, $\boldsymbol{F}$ a set of function symbols, $\boldsymbol{S}$ a set of sort symbols, and ζ is a function that maps each symbol $T \in \boldsymbol{R} \cup \boldsymbol{F}$ to a tuple $\zeta(T) \in \boldsymbol{S}^{\text{arty}(T)}$. We call the tuple $\zeta(T)$ the *type* of T.

We say τ is *relational* if $\boldsymbol{F} = \emptyset$. We call a set X a τ*-set* if $X = \uplus_{s \in \boldsymbol{S}} X_s$ where each X_s is a non-empty set. In other words X is a τ-set if X is the universe of some τ-structure. We can view a vocabulary as a many-sorted vocabulary with a single sort.

2.3.2 Structures

Let $\tau = (\boldsymbol{R}, \boldsymbol{F})$ be a vocabulary. A τ-structure $\mathcal{A} = (A, (R^{\mathcal{A}})_{R \in \boldsymbol{R}}, (F^{\mathcal{A}})_{F \in \boldsymbol{F}})$ consists of a non-empty set A called the *domain* or *universe* of $\mathcal{A}$ together with relations $R^{\mathcal{A}} \subseteq A^{r_R}$ for each $R \in \boldsymbol{R}$ and functions $F^{\mathcal{A}} : A^{f_F} \to A$ for each $F \in \boldsymbol{F}$. The *size* or *cardinality* of $\mathcal{A}$ is the cardinality of the universe A and is denoted by $|\mathcal{A}|$. We write $U(\mathcal{A})$ to refer to the universe of $\mathcal{A}$. We use $\mathcal{A}, \mathcal{B}, \ldots$ to denote structures and $A, B, \ldots$ to denote their universes, respectively. We write $\text{fin}[\tau]$ to denote the set of all finite τ-structures. For $n \in \mathbb{N}$ we write $\text{fin}[\tau, n]$ to denote the set of all finite τ-structures of size n. For a finite set A we write $\text{fin}[\tau, A]$ to denote the set of τ-structures with universe A. In this thesis all structures are assumed to be finite unless explicitly stated otherwise.

Let $\tau = (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be a many-sorted vocabulary. A τ-structure

$$\mathcal{A} = (\biguplus_{s \in \boldsymbol{S}} A_s, (R^{\mathcal{A}})_{R \in \boldsymbol{R}}, (F^{\mathcal{A}})_{F \in \boldsymbol{F}})$$

is defined such that for each $s \in \boldsymbol{S}$ we have that A_s is a non-empty set, for each $R \in \boldsymbol{R}$ we have $R^{\mathcal{A}} \subseteq A_{\zeta(R)(1)} \times \ldots \times A_{\zeta(R)(r_R)}$, and for each $F \in \boldsymbol{F}$ we have $F^{\mathcal{A}} : A_{\zeta(F)(1)} \times \ldots \times A_{\zeta(F)(f_F)} \to A$.

For a relational vocabulary (many-sorted or otherwise) τ and a τ-set X the *complete structure on* X is the τ-structure with universe X and each relation being trivial. We denote this structure by $K_{\tau,X}$.

2.3.3 First-Order Logic

Let $\tau = (\boldsymbol{R}, \boldsymbol{F})$ be a vocabulary. First-order logic (FO) is defined as follows. The set of $\text{FO}[\tau]$-*terms* is the smallest set containing all (first-order) variables and such that if $F \in \boldsymbol{F}$ and $t_1, \ldots, t_{f_F}$ are terms then $F(t_1, \ldots, t_{f_F})$ is a term. The set of $\text{FO}[\tau]$-*formulas* is the smallest set containing the atomic formulas $s_1 = s_2$ and $R(t_1, \ldots, t_{r_R})$ where $R \in \boldsymbol{R}$ and $s_1, s_2, t_1, \ldots, t_{r_R}$ are $\text{FO}[\tau]$-terms, and is closed under taking conjunctions, disjunctions as well as under applications of universal and existential quantifiers. Let $\text{FO}[\tau]$ be the set of $\text{FO}[\tau]$-formulas.

We use lower-case letters $x, y, \ldots$ to denote first-order variables. We also allow for first-order formulas that contain second-order variables. We use upper-case letters $V, W, X, Y, \ldots$ to denote second-order variables. Each second-order variable V is associated with an arity $\text{arty}(V) \in \mathbb{N}_0$. We write $\text{free}(\phi)$ denote the free variables of the FO-formula ϕ. Let $\vec{x}$ be a vector of first-order variables and $\vec{V}$ a vector of second-order variables. We write $\phi(\vec{x}; \vec{V})$

to denote that the free first-order variables that appear free in ϕ are among $\vec{x}$ and the second-order variables that appear free in ϕ are among $\vec{V}$. We always assume that the variables in a sequence of first-order or second-order variables are distinct.

The *width* of a formula ϕ is the maximum number of variables that appear free in any particular subformula of ϕ. We write $\mathrm{cl}(\phi)$ to denote the set of all subformulas of ϕ. For a more careful definition of these terms see [36]

In this chapter we define a number of extensions of fixed-point logic. The notation used here extends to these logics as well.

2.3.4 Assignments and Models

Let τ be a vocabulary and let $\mathcal{A}$ be a τ-structure. We say α is an *assignment in* $\mathcal{A}$ if α is a function with domain X, where X is a set of first-order and second-order variables, and is such that for each first-order variable $x \in X$ we have $\alpha(x) \in A$ and for each second-order variable $V \in X$ we have $\alpha(V) \subseteq A^{\mathrm{arty}(V)}$. If we would particularly like to emphasise the domain of α we instead say that α is an *assignment to* X *in* $\mathcal{A}$. For a sequence of variables $\vec{x} = (x_1, \ldots, x_n)$ and a sequence $(a_1, \ldots, a_n)$ in A we write $\frac{\vec{a}}{\vec{x}}$ to denote the assignment that maps x_i to a_i for each $i \in [n]$. For an assignment α with domain X we let $\alpha\frac{\vec{a}}{\vec{x}}$ denote the assignment with domain $X \cup \vec{x}$ such that if $y = x_i$ for some $i \in [n]$ then y is mapped to a_i and otherwise y is mapped to $\alpha(y)$.

Let L be a logic. Let $\mathcal{A}$ be a τ-structure and let α be an assignment in $\mathcal{A}$. We write $\mathcal{A} \models_L \phi[\alpha]$ to denote that $\mathcal{A}$ satisfies ϕ under the assignment α. This definition, of course, depends on the particular logic chosen. If the logic is obvious from context we omit the subscript and just write $\mathcal{A} \models \phi[\alpha]$.

Let $\phi(\vec{x}; \vec{V})$ be a formula where $\vec{x} = (x_1, \ldots x_n)$ and $\vec{V} = (V_1, \ldots, V_m)$. For each $i \in [n]$ let $a_i \in A$ and for each $j \in [m]$ let $R_j \subseteq A^{\mathrm{arty}(V_j)}$. We write $\mathcal{A} \models_L \phi[\vec{a}; \vec{R}]$ to abbreviate $\mathcal{A} \models_L \phi[\alpha]$ where $\alpha(x_i) = a_i$ for all $i \in [n]$ and $\alpha(V_j) = R_j$ for all $j \in [m]$.

Let $\phi(\vec{x}; \vec{V})$ be a formula where $\vec{x} = (x_1, \ldots x_n)$ and $\vec{V} = (V_1, \ldots, V_m)$. Let $\vec{x}^1 = (x_{i_1}, \ldots, x_{i_k})$ be a subsequence of $\vec{x}$ and let $\vec{x}^2 = (x_{j_1} \ldots, x_{j_l})$ be the subsequence of $\vec{x}$ consisiting of all those variables that do not appear in $\vec{x}^1$. Let α be an assignment to $\vec{x}^1 \cup \vec{V}$ in $\mathcal{A}$. Let $\phi^{(\mathcal{A},\alpha)} := \{\vec{a} \in A^l : \mathcal{A} \models \phi[\alpha\frac{\vec{a}}{\vec{x}^2}]\}$. We write $\phi^{\mathcal{A}}$ to denote $\phi^{(\mathcal{A},\alpha)}$ when α is the empty function. We notice that if every free variable in ϕ appears in the domain of α then $\phi^{(\mathcal{A},\alpha)} = \{\emptyset\}$ if $\mathcal{A} \models \phi[\alpha]$ and otherwise $\phi^{(\mathcal{A},\alpha)} = \emptyset$.

Let $t(\vec{x})$ be a term and α be an assignment to $\vec{x}$ in $\mathcal{A}$. We write $t^{(\mathcal{A},\alpha)}$ to denote the value of t for the assignment α.

2.3.5 Fixed-Point Logic

In this thesis we are particularly interested in fixed-point logics and their extensions. We now define the inflationary fixed-point of a formula. For a more detailed introduction please see Libkin [36], Immerman [32], or Grädel et al. [20].

Let τ be a vocabulary. Let $\phi(\vec{x}; V)$ be a τ-formula where $\vec{x}$ is a k-ary tuple of first-order variables and V is a k-ary second-order variable. Let $\mathcal{A}$ be a τ-structure and let α be an assignment in $\mathcal{A}$. We inductively define a sequence of k-ary relations as follows:

$$X_0 := \emptyset$$
$$X_{i+1} := X_i \cup \{\vec{a} \in A^k : \mathcal{A} \models \phi[\alpha \frac{\vec{a}}{\vec{x}} \frac{X_i}{V}]\}.$$

We notice that if $\mathcal{A}$ has size n then for all $j \geq n^k$ we have that $X_j = X_{n^k}$. In other words, this sequence converges to a limit after at most n^k stages. Let $\text{ifp}(\mathcal{A}, \alpha, \phi) \subseteq A^k$ be the limit of this sequence.

We now define (inflationary) *fixed-point logic* (FP). This logic is defined from FO by extending formula formation rules with the following:

> Let $\phi(\vec{x}; V)$ be an FP-formula where $\vec{x}$ is a k-ary tuple of first-order variables and V is a k-ary second-order variable. Let $\vec{t}$ be a k-tuple of terms. Then $[\mathbf{ifp}_{V,\vec{x}}\phi](\vec{t})$ is a formula with free variables $(\text{free}(\phi) \setminus (\vec{x} \cup \{V\})) \cup \bigcup_{i \in [k]} \text{free}(t_i)$.

The semantics for FP is defined by extending the semantic rules of FO such that

$$\mathcal{A} \models [\mathbf{ifp}_{V,\vec{x}}\phi](\vec{t})[\alpha] \text{ if, and only if, } (t_1^{(\mathcal{A},\alpha)}, \ldots, t_k^{(\mathcal{A},\alpha)}) \in \text{ifp}(\mathcal{A}, \alpha, \phi),$$

for any structure $\mathcal{A}$ and assignment α to X in $\mathcal{A}$, where X is a set of variables containing all variables that appear free in $[\mathbf{ifp}_{V,\vec{x}\vec{\mu}}\phi](\vec{t})$.

2.3.6 Logics with a Number Sort

In this subsection we define two-sorted extensions of first-order and fixed-point logic with a number sort. For an a more detailed introduction please see [29, 38].

We first define *first-order logic with a number sort* ($\text{FO}^{\mathbb{N}}$). This logic is a two-sorted extension of FO. Let τ be a vocabulary that does not contain any symbol in $\{\leq, +, \cdot, 0_N, 1_N\}$. Each first-order variable in $\text{FO}^{\mathbb{N}}$ is either an *element variable* or a *number variable*. The element variables range over the domain of the structure and the number variables over a corresponding number domain. We use Latin lower-case letters $x, y, \ldots$ to denote element variables and Greek lower-case letters $\mu, \nu, \ldots$ to denote number variables. The formulas of $\text{FO}^{\mathbb{N}}$ may contain mixed-sort second-order variables. This means that each second-order

variable V is associated with a *type* $(k_1, k_2) \in \mathbb{N}_0^2$ such that $k_1 + k_2$ is equal to the arity of V. Let $\mathcal{A}$ be a τ-structure. We say that a function α is *an assignment to X in $\mathcal{A}$* if X is a set consisting of element and number variables and mixed-sort second-order variables such that for all $x \in X$ if x is an element variable then $\alpha(x) \in A$, if x is a number variable then $\alpha(x) \in [|\mathcal{A}|]_0$, and for every second-order variable $V \in X$ of type (k_1, k_2) we have $\alpha(V) \subseteq A^{k_1} \times [|\mathcal{A}|]_0^{k_2}$.

We now give a brief description of the logic. Since the logic is two sorted there are two sorts of terms, *element terms* (which we often just call terms) and *number terms*. The set of element terms is equal to the set of FO[τ]-terms. We now define the set of number terms recursively as follows. The constant symbols 0_N and 1_N are number terms. Each number variable is an number term. If s and t are-number terms then $s + t$ and $s \cdot t$ are number terms. The FO$^{\mathbb{N}}$[τ]-formulas are defined by extending the formula formation rules of FO such that if ϕ is an FO$^{\mathbb{N}}$-formula then $\forall\mu\,\phi$ and $\exists\mu\,\phi$ are FO$^{\mathbb{N}}$[τ]-formulas and if s and t are number terms then $s = t$ and $s \leq t$ are FO$^{\mathbb{N}}$[τ]-formulas.

The semantics of FO$^{\mathbb{N}}$ is defined by extending the semantic rules for FO as follows. Let $\mathcal{A}$ be a τ-structure and α be an assignment in $\mathcal{A}$. Let s be a number term. We now define the semantics of s for a structure $\mathcal{A}$ and assignment α, which we denote by $s^{(\mathcal{A},\alpha)}$, recursively as follows. If $s = 0_N$ then $s^{(\mathcal{A},\alpha)} = 0$ and if $s = 1_N$ then $s^{(\mathcal{A},\alpha)} = 1$. If s is some number variable μ then $s^{(\mathcal{A},\alpha)} = \alpha(\mu)$. If $* \in \{\cdot, +\}$ then if $s = s_1 * s_2$ then $s^{(\mathcal{A},\alpha)} = s_1^{(\mathcal{A},\alpha)} * s_2^{(\mathcal{A},\alpha)}$. If $\phi \equiv s = t$ then $\mathcal{A} \models \phi[\alpha]$ if, and only if, $s^{(\mathcal{A},\alpha)} = t^{(\mathcal{A},\alpha)}$. If $\phi \equiv \forall\mu\,\psi$ then $\mathcal{A} \models \phi[\alpha]$ if, and only if, for every $a \in [|\mathcal{A}|]_0$ we have $\mathcal{A} \models \psi[\alpha\frac{a}{\mu}]$. If $\phi \equiv \exists\mu\,\psi$ then $\mathcal{A} \models \phi[\alpha]$ if, and only if, there exists $a \in [|\mathcal{A}|]_0$ such that $\mathcal{A} \models \psi[\alpha\frac{a}{\mu}]$.

Remark 2.1. It follows from the above definition that the evaluation of a number variable in a structure of size n is always an element of the set $\{0, \ldots, n\}$. We can use sequences of number variables in order to encode larger numbers in base $n + 1$. With this in mind, when defining a number term we often abuse notation and write $\vec{\mu}$ to abbreviate $\Sigma_{i\in[|\vec{\mu}|]}\mu_i(\epsilon+1)^{i-1}$, where ϵ is a number term evaluating to the size of the universe. For example, if t is a number term we may write $\vec{\mu} \leq t$ to denote the formula $\exists\epsilon\,[\forall\lambda\,(\lambda \leq \epsilon) \wedge ((\Sigma_{i\in[|\vec{\mu}|]}\mu_i \cdot (\epsilon+1)^{i-1}) \leq t)]$, where ϵ is a number variable that does not appear free in t.

We now define *fixed-point logic with a number sort* (FP$^{\mathbb{N}}$). This logic is defined by extending FO$^{\mathbb{N}}$ with a fixed-point operator in a similar way as we defined FP by extending FO with a fixed-point operator. In this case we allow for mixed-sort second-order variables. Given the central importance of this logic in this thesis we now explicitly define the inflationary fixed-point of a formula with a number-sort and then formally define FP$^{\mathbb{N}}$.

Let τ be a vocabulary. Let $\phi(\vec{x}, \vec{\mu}; V)$ be a τ-formula where $\vec{x}$ is a k_1-ary tuple of element-sort first-order variables, $\vec{\mu}$ is a k_2-ary tuple of number-sort first-order variables, and V is a second-order variable with type (k_1, k_2). Let $\mathcal{A}$ be a τ-structure and let α be an assignment in $\mathcal{A}$. We inductively define a sequence of $(k_1 + k_2)$-ary relations as follows:

$$X_0 := \emptyset$$
$$X_{i+1} := X_i \cup \{(\vec{a}, \vec{m}) \in A^{k_1} \times [|\mathcal{A}|]_0^{k_2} : \mathcal{A} \models \phi[\alpha \frac{\vec{a}}{\vec{x}} \frac{\vec{m}}{\vec{\mu}} \frac{X_i}{V}]\}.$$

Let $N = n^{k_1}(n+1)^{k_2}$. We notice that if $\mathcal{A}$ is a structure of size n then for all $j \geq N$ we have that $X_j = X_N$. It follows that this sequence converges to a limit after at most n^k steps. Let $\text{ifp}(\mathcal{A}, \alpha, \phi) \subseteq A^{k_1} \times [|\mathcal{A}|]_0^{k_2}$ be the limit of this sequence.

We now are now ready to formally define $\text{FP}^{\mathbb{N}}$. This logic is defined from $\text{FO}^{\mathbb{N}}$ by extending formula formation rules with the following:

> Let $\phi(\vec{x}, \vec{\mu}; V)$ be an $\text{FP}^{\mathbb{N}}$-formula where $\vec{x}$ is a k_1-length tuple of element-sort first-order variables, $\vec{\mu}$ is a k_2-length tuple of number-sort first-order variables and, V is a second-order variable of type (k_1, k_2). Let $\vec{t^1}$ be a k_1-tuple of element terms and $\vec{t^2}$ be a k_2-tuple of number-terms. Let $\vec{t}$ be the concatenation of the tuples $\vec{t^1}$ and $\vec{t^2}$. Then $[\mathbf{ifp}_{V,\vec{x}\vec{\mu}}\phi](\vec{t})$ is a formula with free variables $(\text{free}(\phi) \setminus (\vec{x} \cup \vec{\mu} \cup \{V\})) \cup \bigcup_{i \in [k]} \text{free}(t_i)$.

The semantics for $\text{FP}^{\mathbb{N}}$ is defined by extending the semantic rules of FO such that

$$\mathcal{A} \models [\mathbf{ifp}_{V,\vec{x}\vec{\mu}}\phi](\vec{t})[\alpha] \text{ if, and only if, } (t_1^{(\mathcal{A},\alpha)}, \ldots, t_k^{(\mathcal{A},\alpha)}) \in \text{ifp}(\mathcal{A}, \alpha, \phi),$$

for any structure $\mathcal{A}$ and assignment α to X in $\mathcal{A}$, where X is a set of variables containing all variables that appear free in $[\mathbf{ifp}_{V,\vec{x}\vec{\mu}}\phi](\vec{t})$.

2.3.7 Logics with Counting

In this subsection we briefly define logics with both a number-sort and a counting operator that allows them to define the cardinality of a definable set. For a detailed introduction to these logics please see [38].

We define *fixed-point logic with counting* (FPC) by extending $\text{FP}^{\mathbb{N}}$ with the following formula formation rule:

> Let ϕ be an FPC-formula. Let $\vec{x}$ be a sequence of element variables and $\vec{\mu}$ be a sequence of number variables. Then $\#_{\vec{x}\vec{\mu}}\phi$ is a number term with free variables $\text{free}(\phi) \setminus (\vec{x} \cup \vec{\mu})$.

The semantics of FPC is defined by extending the semantic rules for $\text{FP}^{\mathbb{N}}$ such that for a number-term of the form $s = \#_{\vec{x}\vec{\mu}}\phi$, a structure $\mathcal{A}$, and an assignment α to the free variables of s in $\mathcal{A}$ we have

$$s^{(\mathcal{A},\alpha)} = |\{(\vec{a}, \vec{m}) \in A^{|\vec{x}|} \times [|\mathcal{A}|]_0^{|\vec{m}|} : \mathcal{A} \models \phi[\alpha \frac{\vec{a}}{\vec{x}} \frac{\vec{m}}{\vec{\mu}}]\}|.$$

We can define *first-order logic with counting* (FOC) from $\mathrm{FO}^{\mathbb{N}}$ by similarly extending $\mathrm{FO}^{\mathbb{N}}$ with a counting operator. We omit the details here and direct the reader to [38] for a more detailed discussion.

2.3.8 Logics with Rank

We define *fixed-point logic with rank* (FPR) by extending the formula formation rules of $\mathrm{FP}^{\mathbb{N}}$ as follows:

> Let η be an FPR-formula. Let $\vec{x}$ and $\vec{y}$ be sequences of element variables and let $\vec{\mu}$ and $\vec{\nu}$ be sequences of number variables. Let π be a number term. Then $\mathbf{rk}[\pi][(\vec{x}\vec{\mu}, \vec{y}\vec{\nu})\,\eta]$ is a number term with free variables $\mathrm{free}(\pi) \cup \mathrm{free}(\eta) \setminus (\vec{x} \cup \vec{y} \cup \vec{\mu} \cup \vec{\nu})$.

The semantics of FPR is defined by extending the semantic rules for $\mathrm{FP}^{\mathbb{N}}$ as follows. Let $s := \mathbf{rk}[\pi][(\vec{x}\vec{\mu}, \vec{y}\vec{\nu})\,\phi]$ and let $\mathcal{A}$ be a structure and α be an assignment to the free variables of s in $\mathcal{A}$. If $\pi^{(\mathcal{A},\alpha)}$ is not prime let $s^{(\mathcal{A},\alpha)} = 0$. Otherwise let $p := \pi^{(\mathcal{A},\alpha)}$, $I := A^{|\vec{x}|} \times [|\mathcal{A}|]_0^{|\vec{\mu}|}$, $J := A^{|\vec{y}|} \times [|\mathcal{A}|]_0^{|\vec{\mu}|}$, and $M : I \times J \to \mathbb{F}_p$ be defined such that $M(\vec{a}\vec{m}, \vec{b}\vec{n}) = \eta^{(\mathcal{A},\beta)} \mod p$ where $\beta := \alpha \frac{\vec{a}}{\vec{x}} \frac{\vec{m}}{\vec{\mu}} \frac{\vec{b}}{\vec{y}} \frac{\vec{n}}{\vec{\nu}}$ for each $\vec{a}\vec{m} \in I$ and $\vec{b}\vec{n} \in J$. Let $s^{(\mathcal{A},\alpha)}$ be the rank of M understood as a matrix over the finite field with characteristic p.

We can define *first-order logic with rank* by similarly extending the formula formation rules of $\mathrm{FO}^{\mathbb{N}}$ with rank operators as above.

We should note that this formulation of a rank logic is due to Grädel and Pakusa [22]. The first rank logic was introduced by Dawar et al. [13]. The logic they introduced was defined by extending $\mathrm{FP}^{\mathbb{N}}$ by a family of rank operators each of which computed the rank of the matrix over the field of characteristic p. In contrast the rank logic introduced here extends $\mathrm{FP}^{\mathbb{N}}$ with a single rank operator that takes the characteristic as a parameter. It was shown in [22] that FPR is strictly more expressive than the rank logic introduced by Dawar et al. [13].

2.3.9 Queries and Classes

Let ρ be a vocabulary. Let $\mathcal{C} \subseteq \mathrm{fin}[\rho]$. We say that $\mathcal{C}$ is a class of structures if $\mathcal{C}$ is closed under isomorphism, which means that for each $\mathcal{A}, \mathcal{B} \in \mathrm{fin}[\rho]$ if $\mathcal{A} \in \mathcal{C}$ and $\mathcal{A} \cong \mathcal{B}$ then $\mathcal{B} \in \mathcal{C}$. Let L be a logic and $\phi \in L[\rho]$.

Definition 2.2. (Query [20]) Let $m \in \mathbb{N}$. Let ρ be a vocabulary. Let $\mathcal{C}$ be a class of ρ-structures. An *m-ary query on $\mathcal{C}$* is a function Q with domain $\mathcal{C}$ such that for each $\mathcal{A}$

- $Q(\mathcal{A})$ is an m-ary relation on $\mathcal{A}$ for each $\mathcal{A} \in \mathcal{C}$,
- Q is preserved under isomorphism, i.e. for each $\mathcal{A}, \mathcal{B} \in \mathcal{C}$, if $h : \mathcal{A} \to \mathcal{B}$ is an isomorphism then $Q(\mathcal{B}) = h(Q(\mathcal{A}))$.

A *Boolean query* on a class $\mathcal{C}$ is a mapping $Q : \mathcal{C} \to \{0, 1\}$ that is preserved under isomorphism. We can identify each Boolean query with a class of structures $\mathcal{C}_Q := \{\mathcal{A} \in \mathcal{C} : Q(\mathcal{A}) = 1\}$. We omit any mention of $\mathcal{C}$ when $\mathcal{C} = \text{fin}[\rho]$ or $\mathcal{C}$ is clear from context.

Let τ be a vocabulary and let ϕ be a sentence in τ. Let

$$\text{Mod}(\phi) := \{\mathcal{A} \in \text{fin}[\tau] : \mathcal{A} \models \phi\}$$

be the class of (finite) models of ϕ. Let L be a logic. We say a class of structures $\mathcal{C} \subseteq \text{fin}[\tau]$ is *L-definable* if there exists $\phi \in L[\tau]$ such that $\text{Mod}(\phi) = \mathcal{C}$. We say that an m-ary query Q on τ-structures is *L-definable* if there exists $\phi(\vec{x}) \in L[\tau]$ such that $|\vec{x}| = m$ and for any $\mathcal{A} \in \text{fin}[\tau]$ we have $Q(\mathcal{A}) = \{\vec{a} \in A^m : \mathcal{A} \models \phi[\vec{a}]\}$.

Let L_1 and L_2 be logics. We say that L_1 is *at least as expressive as* L_2 if every L_2-definable query is L_1-definable. In this case we write $L_2 \leq L_1$. If $L_1 \leq L_2$ and $L_2 \leq L_1$ we write $L_1 \equiv L_2$.

2.3.10 Interpretations

Let L be a logic. Let ρ and τ be relational vocabularies and let $\vec{w}$ be a sequence of variables. A *$L[\rho, \tau]$-interpretation* with *width* k and *parameters* $\vec{w}$ is a sequence of $L[\rho]$-formulas $\mathcal{I} := \langle \phi^D, \phi^{\approx}, (\phi_R)_{R \in \rho} \rangle$ such that

- ϕ^D has free variables among $(\vec{x}, \vec{w})$ where $\vec{x}$ is a k-tuple of variables;
- $\phi^{\approx}$ has free variables among $(\vec{x}, \vec{y}, \vec{w})$ where $\vec{x}$ and $\vec{y}$ are k-tuples of variables;
- for each $R \in \boldsymbol{R}$ the free variables in ϕ_R are among $(\vec{x}_1^R, \ldots, \vec{x}_{r_R}^R, \vec{w})$ where for each $i \in [r_R]$, $\vec{x}_i^R$ is a k-tuple of variables.

We call ϕ^D the *domain* formula and $\phi^{\approx}$ the *equality* formula. Let $\mathcal{A} \in \text{fin}[\rho]$ and let α be an assignment to $\vec{w}$ in $\mathcal{A}$. Let $\mathcal{B}$ be a τ-structure. We say that $\mathcal{I}$ *interprets* $\mathcal{B}$ *in* $(\mathcal{A}, \alpha)$ (and write $\mathcal{B} = \mathcal{I}(\mathcal{A}, \alpha)$) if there exists a surjection $h : (\phi^D)^{(\mathcal{A},\alpha)} \to B$, called *the coordinate map*, such that

- for all $\vec{a}, \vec{b} \in (\phi^D)^{(\mathcal{A},\alpha)}$ we have

$$\mathcal{A} \models \phi^{\approx}[\alpha \frac{\vec{a}}{\vec{x}} \frac{\vec{b}}{\vec{y}}] \iff h(\vec{a}) = h(\vec{b}) \text{ and}$$

- for all $R \in \rho$, $i \in [r_R]$, and $\vec{a}_i \in (\phi^D)^{(\mathcal{A},\alpha)}$ we have

$$(\vec{a}_1, \ldots, \vec{a}_{r_R}) \in \phi_R^{(\mathcal{A},\alpha)} \iff (h(\vec{a}_1), \ldots, h(\vec{a}_{r_R})) \in R^{\mathcal{B}}$$

We say that $\mathcal{I}(\mathcal{A}, \alpha)$ is *defined* if there exists a τ-structure $\mathcal{B}$ such that $\mathcal{B} = \mathcal{I}(\mathcal{A}, \alpha)$. It follows that $\mathcal{I}$ interprets $\mathcal{B}$ in $(\mathcal{A}, \alpha)$ if, and only if, the binary relation $\approx$ defined by $\phi^{\approx}$ is a congruence

on the structure $((\phi^D)^{(\mathcal{A},\alpha)}, (\phi_R^{(\mathcal{A},\alpha)})_{R\in\rho})$ and there exists a function $h : (\phi^D)^{(\mathcal{A},\alpha)} \to B$ such that h is an isomorphism from the quotient structure $((\phi^D)^{(\mathcal{A},\alpha)}, (\phi_R^{(\mathcal{A},\alpha)})_{R\in\rho})/\approx$ to $\mathcal{B}$.

2.3.11 Lindström Quantifiers

The generalised quantifiers introduced by Lindström provide a mechanism for studying extensions of a logic that add the ability to express a particular query [37]. Let ρ be a relational vocabulary. Let $\mathcal{G}$ be a class of ρ-structures. We associate $\mathcal{G}$ with a *Lindström quantifier* $Q_\mathcal{G}$. For a logic L the extension $L(Q_\mathcal{G})$ is defined by extending the definition of L with the following formula formation rule:

> For each $R \in \rho$ let ϕ_R be an $L(Q_\mathcal{G})$-formula and let $\vec{x}^R := (x_1^R, \ldots, x_{r_R}^R)$ be a sequence of variables. Let $\phi^D(x, \vec{w})$ and $\phi^{\approx}(x, y, \vec{w})$ be $L(Q_\mathcal{G})$-formulas. Then $Q_\mathcal{G}[\phi^D, \phi^{\approx}][\vec{x}^R\, \phi_R]_{R\in\rho}$ is an $L(Q_\mathcal{G})$-formula with free variables $(\text{free}(\phi^D) \setminus \{x\}) \cup (\text{free}(\phi^{\approx}) \setminus \{x, y\}) \cup \bigcup_{R\in\rho}(\text{free}(\phi_R) \setminus \vec{x}_R)$.

The semantics of $L(Q_\mathcal{G})$ is defined by extending the semantics of L such that for any finite structure $\mathcal{A}$ and an assignment α in $\mathcal{A}$ we have that if $\mathcal{I} = \langle \phi^D, \phi^{\approx}, (\phi_R)_{R\in\rho}\rangle$ then

$$\mathcal{A} \models Q_\mathcal{G}[\phi^D, \phi^{\approx}][\vec{x}^R\, \phi_R]_{R\in\rho}[\alpha] \text{ if, and only if, } \mathcal{I}(\mathcal{A}, \alpha) \text{ is defined and } \mathcal{I}(\mathcal{A}, \alpha) \in \mathcal{G}$$

Example 2.3. Let $\rho = \{U\}$ where U is a unary relation. Let $i \in \mathbb{N}$ and $\mathcal{G}_i$ be the class of ρ-structures such that for all $\mathcal{A} \in \text{fin}[\rho]$, $\mathcal{A} \in \mathcal{G}_i$ if, and only if, $|U^\mathcal{A}| \geq i$. Then the Lindström quantifier $Q_{\mathcal{G}_i}$ is the *counting quantifier* usually denoted by $\exists^{\geq i}$. For an introduction to counting quantifiers see [38].

2.3.12 Infinitary Logics

We also define infinitary logics. We direct the reader to [38] for a more detailed discussion of these logics. For $k \in \mathbb{N}$ let FO^k be the fragment of FO where each formula only use the variables $x_1, \ldots, x_k$. Let $\mathcal{L}^k$ be the closure of FO^k under infinite conjunctions and disjunctions. Let $\mathcal{L}^\omega := \bigcup_{k\leq\omega} \mathcal{L}^k$.

We can define an extension for each of these finite-variable infinitary logics by counting quantifiers. We let FO+C be the extension of FO with the counting quantifiers mentioned in Example 2.3. For each $k \in \mathbb{N}$ let FO+C^k be the fragment of FO+C where each formula only uses the variables $x_1, \ldots, x_k$. Let $\mathcal{C}^k$ be the corresponding infinitary logic and let $\mathcal{C}^\omega$ be the union of these logics.

There is an interesting relationship between these infinitary logics and fixed-point logics. It can be shown that $\text{FP} \leq \mathcal{L}^\omega$ [35] and $\text{FPC} \leq \mathcal{C}^\omega$ (see [38] for details).

2.4 Complexity Theory and Logic

In this section we review common concepts in complexity theory and descriptive complexity theory. We direct the reader to [4] for a more thorough introduction to complexity theory and to [32] for a thorough introduction to descriptive complexity theory.

2.4.1 Basic Notions and Complexity Classes

We write P to denote the set of languages decidable by a deterministic Turing machine with a polynomial bound on its running time. We write NP to denote the set of languages decidable by a non-deterministic Turing machine with a polynomial bound on its running time.

2.4.2 Capturing Complexity Classes

We now review some basic definitions and results in descriptive complexity. We note that there is a natural way of encoding a relational structure as a binary string. We first recall that for each order on the vertices of a graph G there is a binary encoding of G by its adjacency matrix. It follows that we can associate a graph G with the set of all possible encodings, $\text{enc}(G)$, one for each possible order on the vertices of G. For a class of graphs $\mathcal{C}$ we can define a corresponding a language $\text{enc}(\mathcal{C}) := \cup_{G \in \mathcal{C}} \text{enc}(G)$. This approach to encoding graphs can be generalised in an obvious way to finite relational structures. For more details see [36]. In this way we can define for each class of relational structures $\mathcal{C}$ over a relational vocabulary ρ a language $\text{enc}(\mathcal{C}) \subseteq \{0,1\}^*$. We say that $\mathcal{C}$ is in P if $\text{enc}(\mathcal{C}) \in \text{P}$.

We will formally state the question of whether there is a logic for P. In order to do so we need a formal definition of a logic.

Definition 2.4. A *logic* L consists of:

- a decidable set $L[\rho]$, whose elements we call L-*sentences*, for each relational vocabulary ρ; and
- a binary relation $\models_L$ between finite structures and L-sentences such that for each vocabulary ρ and each $\phi \in L[\rho]$ the set $\{\mathcal{A} \in \text{fin}[\rho] : \mathcal{A} \models_L \phi\}$ is closed under isomorphism.

We now formally define the notion of a logic capturing polynomial-time. We informally understand it as asserting that such a logic must define exactly those classes of structures that are decidable in P, with the additional requirement that the function that maps a formula of the logic to a Turing machine computing the corresponding algorithm must be computable. This formulation is due to Gurevich [26].

Definition 2.5. We say a logic L captures P if for every vocabulary ρ

- every Boolean query decidable in P is definable in L;

- there is a computable function that associates with every L-sentence $\phi \in L[\rho]$ a pair $(p(x), M)$ where $p(x)$ is a polynomial and M is a Turing machine such that M decides mod ϕ in time $p(n)$

We now state the Gurevich conjecture [26].

Conjecture 2.6. *There is no logic that captures* P*.*

We lastly note a fundamental result proved Immerman and Vardi [30, 42] that establishes that over ordered structures any polynomial-time decidable query is FP-definable.

Theorem 2.7 (Immerman-Vardi Theorem)**.** *Let ρ be a relational vocabulary such that there is some binary relation symbol $\leq$ in ρ. Let $\mathcal{C}$ be a class of ρ-structures such that for each $\mathcal{A} \in \mathcal{C}$, $\leq^{\mathcal{A}}$ is a linear order on the universe of $\mathcal{A}$. Let Q be a query on $\mathcal{C}$. Then Q is polynomial-time decidable if, and only if, Q is* FP*-definable.*

2.4.3 Vectorised Families of Quantifiers

We review the notion of a *vectorised family of quantifiers* introduced by Dawar [11] and discuss a result, also proved by Dawar, establishing that if there is a logic that captures P then there an extension of FO by a vectorised family of quantifiers that captures P.

Definition 2.8. Let ρ be a relational vocabulary. Let $\mathcal{C}$ be a class of ρ-structures. Let $n \in \mathbb{N}$. Let U_n be a relation symbol of arity n and for $R \in \rho$ let R_n be a distinct relation symbol with arity $r_R \cdot n$. Let $\rho_n = \{U_n\} \cup \{R_n : R \in \rho\}$. Let $\mathcal{C}_n$ be the class of ρ_n-structures such that $\mathcal{A} \in \mathcal{C}_n$ if, and only if, there exists $\mathcal{A}^* \in \mathcal{C}$ such that $\mathcal{A}^*$ has universe $U_n^{\mathcal{A}}$ and for each $R \in \rho$,

$$R^{\mathcal{A}^*} = \{(\vec{a}_1, \ldots, \vec{a}_{r_R}) : \text{ for each } j \in [r_R],\ \vec{a}_j = (a_{j,1}, \ldots, a_{j,n}) \in U_n^{\mathcal{A}} \text{ and } (a_{1,1}, \ldots, a_{r_R,n}) \in R_n^{\mathcal{A}}\}.$$

For each $n \in \mathbb{N}$ let Q_n be the Lindström quantifier associated with $\mathcal{C}_n$. We call the set $\mathbf{Q} = \{Q_n : n \in \mathbb{N}\}$ a *vectorised family of quantifiers* (or just a *vectorised quantifier*) and say that $\mathbf{Q}$ is *generated* by $\mathcal{C}$.

Theorem 2.9 (Dawar [11])**.** *There is a logic that captures* P *if, and only if, there exists a vectorised quantifier $\mathbf{Q}$ such that* FO($\mathbf{Q}$) *captures* P*.*

2.5 Circuits and Logic

2.5.1 Boolean Functions

A (finite) *Boolean function* is a function of the form $f : \{0,1\}^X \to \{0,1\}$, for some finite index set X. We call a Boolean function $f : \{0,1\}^X \to \{0,1\}$ *symmetric* if for all $\sigma \in \mathbf{Sym}_X$

and $\vec{a} \in \{0,1\}^X$, $f(\vec{a}\sigma) = f(\vec{a})$. In other words a Boolean function is symmetric if, and only if, the output of the function is entirely determined by the number of ones in the input. We say a Boolean function $f : \{0,1\}^* \to \{0,1\}$ is *symmetric* if each restriction to strings of size n is symmetric for all $n \in \mathbb{N}$. For a more detailed discussion of Boolean functions we direct the reader to [43] and [4].

2.5.2 Circuits

In this section we briefly introduce Boolean circuits. We direct the reader to [43] and [4] for a more detailed introduction.

A basis is a set of Boolean functions. The *standard basis* $\mathbb{B}_{\text{std}}$ consists of the functions AND, OR, and NOT. The *majority basis* $\mathbb{B}_{\mathbf{maj}}$ is the extension of the standard basis with the majority function MAJ. This majority function evaluates to one if, and only if, at least half the input string consists of ones. Let $n \in \mathbb{N}$. We write AND$[n]$ to denote the restriction of the function AND to the set of all n-length binary strings. In other words, AND$[n] : \{0,1\}^n \to \{0,1\}$ is defined such that for all $\vec{x} \in \{0,1\}^n$ we have AND$[n](\vec{x}) = 1$ if, and only if, there exists $x_i \in \vec{x}$ such that $x_i = 1$. We similarly define OR$[n]$, MAJ$[n]$.

A *Boolean circuit* C of *order* n is a labelled directed acyclic graph (DAG) with a designated set of *input gates* each labelled by a variable $x_1, \ldots, x_n$, and each with in-degree 0, a set of *internal gates* labelled by elements from some Boolean basis, and a single internal gate with out-degree 0 designated as the *output gate*. The evaluation of circuit C of order n for an input $\vec{a} \in \{0,1\}^n$ is denoted by $C[\vec{a}]$ and computed by assigning the input gates accordingly and recursively evaluating the gates in the circuit. We denote the evaluation of a gate g in C for the input $\vec{a}$ by $C[\vec{a}](g)$. If g is an input gate, then $g = x_i$ for some $i \in [n]$, and $C[\vec{a}](g) = a_i$. If g is an internal gate then g is labelled by a symbol denoting a Boolean operation, and $C[\vec{a}](g)$ is the result of applying that operation to the string formed from the evaluations of those gates input to g. If g is the output gate, let $C[\vec{a}] = C[\vec{a}](g)$.

The *size* of a circuit C, denoted by $|C|$, is the number of gates in the circuit. The *depth* of a gate g is the longest path from an input gate to g, and the *depth* if the circuit is the depth of the output gate. The *width* of a circuit is the maximum size of a set of gates with the same depth.

If C is a circuit of order n, then C computes a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$ defined by $f_C(\vec{a}) = C[\vec{a}]$. Let $(C_n)_{n\in\mathbb{N}}$ be a family of circuits, where C_n has order n. We say $(C_n)_{n\in\mathbb{N}}$ *decides* a language $L : \{0,1\}^* \to \{0,1\}$ if for all $\vec{a} \in \{0,1\}^*$, $C_{|\vec{a}|}[\vec{a}] = L(\vec{a})$. We say that a family of circuits $(C_n)_{n\in\mathbb{N}}$ has *polynomial size* if the function that maps $n \mapsto |C_n|$ is bounded by a polynomial in n. We say that a family of circuits is P-*uniform* if the function $n \mapsto C_n$ is computable in time bounded by a polynomial in n.

2.5.3 Circuits on Structures and Symmetric Circuits

The usual definition of a circuit has input gates labelled by variables and takes as input a binary string. A *circuit on structures*, as defined by Anderson and Dawar [3], takes as input an encoding of a τ-structure and outputs a set of tuples from that structure. More formally, for a relational vocabulary τ, a basis $\mathbb{B}$ of symmetric functions, and $q, n \in \mathbb{N}$, a *circuit on structures* is a DAG with (i) a designated set of input gates, each labelled by a relation symbol $R \in \tau$ and a tuple $\vec{a} \in [n]^{\text{arty}(R)}$ or either 0 or 1, (ii) a set of internal gates, each labelled by an element of $\mathbb{B}$, and (iii) a designated set of output gates each labelled by an element of $[n]^q$.

Let $\mathcal{A}$ be a τ-structure of size n and γ be a bijection from the universe of $\mathcal{A}$ to $[n]$. The bijection γ defines an encoding of $\mathcal{A}$ as a τ-structure with universe $[n]$. The input gate labelled by the relation symbol R and tuple $\vec{a}$ is assigned to one if, and only if, $\vec{a}$ is an element of the interpretation of R in the encoding of $\mathcal{A}$. The circuit may then be recursively evaluated, in a similar way as for conventional circuits, and the evaluation of the output gates is taken as the output of the circuit. Since each output gate is labelled by an element of $[n]^q$, the output of the circuit is an element of $\{0,1\}^{[n]^q}$.

A circuit is called *invariant* if its output does not depend on the choice of γ. In this case the circuit of order n defines a q-ary query for structures of size n and a family $(C_n)_{n\in\mathbb{N}}$ of invariant circuits defines a q-ary query. If C is an invariant circuit with $q = 0$, i.e. C defines a Boolean query, then the circuit has a single output gate and C decides a property of τ-structures. A circuit is called *symmetric* if every permutation on the universe $[n]$, each of which induces a permutation on the input gates, extends to an automorphism of the circuit. A symmetric circuit is necessarily invariant. Anderson and Dawar showed that P-uniform families of symmetric circuits over the majority basis have exactly the same expressive power as FPC and that P-uniform families of symmetric circuits over the standard basis have exactly the same expressive power as $\text{FP}^{\mathbb{N}}$ [3].

We direct the reader to [3] for a much more detailed introduction to symmetric circuits.

Chapter 3

Generalised Operators

The study of extensions of fixed-point logics plays an essential role in the field of descriptive complexity. In particular, FPC has been shown to capture polynomial-time over numerous graph classes (see [24]) and has become so central to the field that it is has been described as 'the logic of reference' [12] in descriptive complexity. The logic FPR is one of the strongest logics known to be contained in P but not known to be properly contained, and is a logic of fundamental interest. In this chapter we introduce the notion of a *generalised operator* and show that these operators generalise Lindström quantifiers, counting operators, and rank operators. We use these operators to develop a unified framework for studying extensions of fixed-point logics.

We can think of generalised operators as the natural generalisation of Lindström quantifiers for logics capable of defining number-terms. We recall that a Lindström quantifier is defined by a class of relational structures. An application of a Lindström quantifier is applied to a sequence of formulas which together define a relational structure. The meaning of an application of a quantifier is determined by whether this structure is a member of the associated class or not. In contrast, an application of a generalised operator binds sequences of formulas *and* number-terms which together define a structure with both relations *and* number-valued functions. The meaning of an application of a generalised operator is determined by an *evaluation function* that maps the class of structures with number-valued functions to either a Boolean domain or the natural numbers. These two choices correspond to generalised operators that define number-terms or formulas.

The focus on fixed-point logics and the restriction to finite structures means that most of the results from classical model theory, e.g. the compactness theorem, used for proving inexpressibility results are unavailable. Instead, a standard approach has been to establish a translation from a given fixed-point logic to a bounded-variable infinitary logic. These infinitary logics often admit pebble game characterisations which have proven very useful for establishing inexpressibility results for fixed-point logics (e.g. [9]). The translations from FP to $\mathcal{L}^\omega$ [35] and from FPC to $\mathcal{C}^\omega$ [38] have proved particularly crucial. In this chapter we

generalise these translations for extensions of fixed-point logics by generalised operators. We show that a family of generalised operators $\mathbf{\Omega}$ can be associated with a family of quantifiers $\mathbf{Q}_{\mathbf{\Omega}}$ and that each formula in the extension of $\mathrm{FP}^{\mathbb{N}}$ by $\mathbf{\Omega}$ can be translated to a formula in the extension of $\mathcal{C}^{\omega}$ by $\mathbf{Q}_{\mathbf{\Omega}}$. As an intermediate step we introduce the notion of a *substitution program*. We think of a substitution program as a means of more compactly representing the unrolling of a fixed-point operator. We use substitution-programs again in Chapter 5 when we define a translation from extensions of fixed-point logics to families of symmetric circuits.

This chapter is organised as follows. In Section 3.1 we formally introduce *number-extended structures* and correspondingly generalise the notion of an interpretation. In Section 3.2 we introduce generalised operators and discuss extensions of logics by generalised operators. We also prove many useful normal forms for logics extended by generalised operators. In Section 3.3 we introduce *many-sorted quantifiers*, a generalisation of Lindström quantifiers, and show how to associate a family of generalised operators with a family of many-sorted quantifiers. In Section 3.4 we introduce substitution programs and show that a formula of a fixed-point logic extended by a family of generalised operators can be translated to a P-uniform family of substitution programs. In Section 3.5 we establish a translation from a fixed-point logic extended by a family of generalised operators to a bounded-variable infinitary logic extended by the corresponding family of many-sorted quantifiers.

3.1 Structures with Number-Valued Functions

A Lindström quantifier is defined by a class of single-sorted relational structures. An application of a quantifier $Q_{\mathcal{G}}$ in a formula defines an interpretation $\mathcal{I}$ and evaluates to true if, and only if, the structure defined by the $\mathcal{I}$ is in $\mathcal{G}$. We aim to define a generalised operator using a similar approach. However, unlike quantifiers, generalised operators should be allowed to operate on formulas *and* number terms. As such, we define a generalised operator from a class of many-sorted structures consisting of both relations and number-valued functions. In this section we generalise the notion of a structure and correspondingly generalise the notion of an interpretation.

Definition 3.1. Let $\tau := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be a many-sorted vocabulary. A *number-extended* τ-structure is a structure of the form $\mathcal{A} = (\uplus_{s \in \boldsymbol{S}} A_s, (R^{\mathcal{A}})_{R \in \boldsymbol{R}}, (F^{\mathcal{A}})_{F \in \boldsymbol{F}})$ where

- for each $s \in \boldsymbol{S}$, A_s is a non-empty set;
- for each $R \in \boldsymbol{R}$, $R^{\mathcal{A}} \subseteq A_{\zeta(R)(1)} \times \ldots \times A_{\zeta(R)(r_R)}$; and
- for each $F \in \boldsymbol{F}$, $F^{\mathcal{A}} : A_{\zeta(F)(1)} \times \ldots \times A_{\zeta(F)(f_F)} \to \mathbb{N}_0$.

Let $\mathcal{A}$ and $\mathcal{B}$ be number-extended τ-structures. We write $h : \mathcal{A} \to \mathcal{B}$ to denote a function from the universe of $\mathcal{A}$ to the universe of $\mathcal{B}$ such that (i) h preserves sorts, (ii) for all $R \in \boldsymbol{R}$ and $\vec{a} \in R^{\mathcal{A}}$ we have $h(\vec{a}) \in R^{\mathcal{B}}$, and (iii) for all $F \in \boldsymbol{F}$ and $\vec{a} \in \mathbf{Dom}(F^{\mathcal{A}})$ we have

$F^{\mathcal{A}}(\vec{a}) = F^{\mathcal{B}}(h(\vec{a}))$. In this case we call h a homomorphism. If h is also a bijection and h^{-1} is a homomorphism then we call h an *isomorphism*.

A many-sorted vocabulary may contain 0-arity (i.e. nullary) relation or function symbols. We call 0-arity function symbols *constant symbols*. We follow the convention of using upper-case letters to denote relation and function symbols (we usually use R, F, or T) and lower-case letters to denote constants. Let τ be a many-sorted vocabulary. Let $\mathrm{fin}^{\mathbb{N}}[\tau]$ denote the set of all number-extended τ-structures.

An interpretation consists of a sequence of formulas and defines a relational structure. In the context of logics with a number sort we can generalise the notion of an interpretation so as to include both formulas *and* number-terms. In this case each number-term defines a number-valued function, and as such the interpretation defines a number-extended structure. We call an interpretation of this sort a *number-extended interpretation*.

Definition 3.2. Let L be a logic with a number sort. Let ρ be a relational vocabulary and let $\tau = (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be a many-sorted vocabulary. Let $\mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0$ and let $\vec{w}$ be a sequence of mixed-sort variables. A *number-extended $L[\rho, \tau]$-interpretation* with *dimension* ar and *parameters* $\vec{w}$ is a sequence of $L[\rho]$-formulas $\mathcal{I} := \langle (\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\approx}_s)_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}}, (\eta_F)_{F\in\mathbf{F}}\rangle$ such that:

- for each $s \in \boldsymbol{S}$ the free variables in ϕ^D_s are among $(\vec{x}^s\vec{\mu}^s, \vec{w})$ where $\vec{x}^s$ is an $\mathrm{ar}(s,1)$-tuple of element variables and $\vec{\mu}^s$ is an $\mathrm{ar}(s,2)$-tuple of number variables;
- for each $s \in \boldsymbol{S}$ the free variables in $\phi^{\approx}_s$ are among $(\vec{x}^s_1\vec{\mu}^s_1, \vec{x}^s_2\vec{\mu}^s_2, \vec{w})$, where $\vec{x}^s_1$ and $\vec{x}^s_2$ are $\mathrm{ar}(s,1)$-tuples of element variables and $\vec{\mu}^s_1$ and $\vec{\mu}^s_2$ are $\mathrm{ar}(s,2)$-tuples of number variables;
- for each $R \in \boldsymbol{R}$ the free variables in ϕ_R are among $(\vec{x}^R_1\vec{\mu}^R_1, \ldots, \vec{x}^R_{r_R}\vec{\mu}^R_{r_R}, \vec{w})$ where for each $i \in [r_R]$, $\vec{x}^R_i$ is an $\mathrm{ar}(\zeta(R)(i), 1)$-tuple of vertex variables and $\vec{\mu}^R_i$ is an $\mathrm{ar}(\zeta(R)(i), 2)$-tuple of number variables; and
- for each $F \in \boldsymbol{F}$ the free variable in η_F are among $(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F}, \vec{w})$ where for each $i \in [f_F]$, $\vec{y}^R_i$ is a $\mathrm{ar}(\zeta(F)(i), 1)$-tuple of vertex variables and $\vec{\nu}^F_i$ is an $\mathrm{ar}(\zeta(F)(i), 2)$-tuple of number variables.

We call $(\phi^D_s)_{s\in\boldsymbol{S}}$ the *domain* formulas and $(\phi^{\approx}_s)_{s\in\boldsymbol{S}}$ the *equality* formulas. Let $\mathcal{A} \in \mathrm{fin}[\rho]$ and let α be an assignment to $\vec{w}$ in $\mathcal{A}$. Let $\mathcal{B}$ be a number-extended τ-structure. Let $B = \uplus_{s\in\boldsymbol{S}} B_i$ be the universe of $\mathcal{B}$. We say that $\mathcal{I}$ *interprets* $\mathcal{B}$ *in* $(\mathcal{A}, \alpha)$ (and write $\mathcal{B} = \mathcal{I}(\mathcal{A}, \alpha)$) if there exists a surjection $h : \uplus_s (\phi^D_s)^{(\mathcal{A},\alpha)} \to \uplus_{s\in\boldsymbol{S}} B_s$, called *the coordinate map*, such that

- for all $s \in \boldsymbol{S}$ and $\vec{a}\vec{m} \in (\phi^D_s)^{(\mathcal{A},\alpha)}$ we have $h(\vec{a}\vec{m}) \in B_s$;
- for all $s \in \boldsymbol{S}$ and $\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2 \in (\phi^D_s)^{(\mathcal{A},\alpha)}$

$$(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\phi^{\approx}_s)^{(\mathcal{A},\alpha)} \iff h(\vec{a}_1\vec{m}_1) = h(\vec{a}_2\vec{m}_2);$$

- for all $R \in \boldsymbol{R}$, $i \in [r_R]$, $\vec{a}_i\vec{m}_i \in (\phi^D_{\zeta(R)(i)})^{(\mathcal{A},\alpha)}$

$$(\vec{a}_1\vec{m}_1, \ldots, \vec{a}_{r_R}\vec{m}_{r_R}) \in \phi_R^{(\mathcal{A},\alpha)} \iff (h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}}; \text{ and}$$

- for all $F \in \boldsymbol{F}$, $i \in [f_F]$, $\vec{a}_i\vec{m}_i \in (\phi^D_{\zeta(F)(i)})^{(\mathcal{A},\alpha)}$

$$\eta_F^{(\mathcal{A},\alpha \frac{\vec{a}_1}{\vec{\imath}_1} \frac{\vec{m}_1}{\vec{\nu}_1} \cdots \frac{\vec{a}_{f_F}}{\vec{\imath}_{f_F}} \frac{\vec{m}_{f_F}}{\vec{\nu}_{f_F}})} = F^{\mathcal{B}}(h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{f_F}\vec{m}_{f_F})).$$

We say that $\mathcal{I}(\mathcal{A}, \alpha)$ is *defined* if there exists a structure $\mathcal{B}$ such that $\mathcal{B} = \mathcal{I}(\mathcal{A}, \alpha)$.

Hence $\mathcal{I}$ interprets $\mathcal{B}$ in $(\mathcal{A}, \alpha)$ if, and only if, the binary relation $\approx := \uplus_{s\in\boldsymbol{S}}\phi_s^{\approx}$ is a congruence on the structure $(\uplus_{s\in\boldsymbol{S}}(\phi_s^D)^{(\mathcal{A},\alpha)}, (\phi_R^{(\mathcal{A},\alpha)})_{R\in\boldsymbol{R}}, (\eta_F^{(\mathcal{A},\alpha)})_{F\in\boldsymbol{F}})$ and there exists a function $h : \uplus_{s\in\boldsymbol{S}}(\phi_s^D)^{(\mathcal{A},\alpha)} \to \uplus_{s\in\boldsymbol{S}}B_s$ such that h is an isomorphism from the quotient structure $(\uplus_{s\in\boldsymbol{S}}(\phi_s^D)^{(\mathcal{A},\alpha)}, (\phi_R^{(\mathcal{A},\alpha)})_{R\in\boldsymbol{R}}, (\eta_F^{(\mathcal{A},\alpha)})_{F\in\boldsymbol{F}})/\approx$ to $\mathcal{B}$.

3.2 Generalised Operators

In this section we introduce the notion of a generalised operator and define what it means to extend a logic by a generalised operator. We also introduce some notation and terminology for generalised operators and prove a normal form for logics extended by generalised operators.

Remark 3.3. We have from Lindström a general approach for extending logics without an assumption the syntax of those logics (for a review of this work see [6]). However, in this chapter we restrict our attention to a particular class of logics that have a syntax 'similar to that of first-order logic'. When, in this chapter we say that L is a logic we mean that L is FO, $\mathrm{FO}^{\mathbb{N}}$, FP, or $\mathrm{FP}^{\mathbb{N}}$, or any extension of these logics by any family of Lindström quantifiers, many-sorted quantifiers, or generalised operators (we define the latter two terms in the course of this chapter). Many results proved in this chapter hold more generally, but this restriction suffices for our purposes.

Let $\tau = (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be a many-sorted vocabulary. Let $E : \mathrm{fin}[\tau] \to \mathbb{N}_0$ be closed under isomorphism, i.e. for all $\mathcal{A}, \mathcal{B} \in \mathrm{fin}[\tau]$, if $\mathcal{A} \simeq \mathcal{B}$ then $E(\mathcal{A}) = E(\mathcal{B})$. Let $\mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0$ be a function. We associate with the pair (E, ar) a *number-valued generalised operator* $\Omega_{(E,\mathrm{ar})}$. We say the *vocabulary* of the operator is τ, the *arity* is ar, and the *evaluation function* is E. For a logic L the extension $L(\Omega_{E,\mathrm{ar}})$ is the closure of the set of formulas in L under the following number-term formation rule:

Let $\vec{w}$ be a tuple of mixed-sort variables. For each $s \in \boldsymbol{S}$ let $\phi_s^D(\vec{x}^s\vec{\mu}^s, \vec{w}) \in L(\Omega_{E,\mathrm{ar}})$ let $\phi_s^{\approx}(\vec{x}_1^s\vec{\mu}_1^s, \vec{x}_1^s\vec{\mu}_1^s, \vec{w}) \in L(\Omega_{E,\mathrm{ar}})$. For each $R \in \boldsymbol{R}$ let $\phi_R(\vec{x}_1^R\vec{\mu}_1^R, \ldots, \vec{x}_{r_R}^R\vec{\mu}_{r_R}^R, \vec{w}) \in L(\Omega_{E,\mathrm{ar}})$. For each $F \in \boldsymbol{F}$ let $\eta_F(\vec{y}_1^F\vec{\nu}_1^F, \ldots, \vec{y}_{f_F}^F\vec{\nu}_{f_F}^F, \vec{w})$ be a number term in $L(\Omega_{E,\mathrm{ar}})$. Let $\mathcal{I} := \langle(\phi_s^D)_{s\in\boldsymbol{S}}, (\phi_s^{\approx})_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}}, (\eta_F)_{F\in\boldsymbol{F}}\rangle$ be a number-extended interpretation with dimension ar and parameters $\vec{w}$. Then

$$\gamma(\vec{w}) \equiv \Omega_{E,\mathrm{ar}}[(\phi_s^D)_{s\in\boldsymbol{S}}, (\phi_s^{\approx})_{s\in\boldsymbol{S}}][(\vec{x}_1^R\vec{\mu}_1^R, \ldots, \vec{x}_{r_R}^R\vec{\mu}_{r_R}^R)\,\phi_R]_{R\in\boldsymbol{F}}[(\vec{y}_1^F\vec{\nu}_1^F, \ldots, \vec{y}_{f_F}^F\vec{\nu}_{f_F}^F)\,\eta_F]_{F\in\boldsymbol{F}}$$

is a number term in $L(\Omega_{E,\mathrm{ar}})$. We have

$$\begin{aligned}\mathrm{free}(\gamma) := &\bigcup_{s\in\boldsymbol{S}} (\mathrm{free}(\phi)_s^D \setminus (\vec{x}^s \cup \vec{\mu}^s)) \cup \bigcup_{s\in\boldsymbol{S}} (\mathrm{free}(\phi)_s^{\approx} \setminus (\vec{x}_1^s \cup \vec{\mu}_1^s \cup \vec{x}_1^s \cup \vec{\mu}_1^s)) \cup \\ &\bigcup_{R\in\boldsymbol{R}} (\mathrm{free}(\phi_R) \setminus (\vec{x}_1^R \cup \vec{\mu}_1^R \cup \ldots \cup \vec{x}_{r_R}^R \cup \vec{\mu}_{r_R}^R)) \cup \\ &\bigcup_{F\in\boldsymbol{F}} (\mathrm{free}(\phi_F) \setminus (\vec{x}_1^F \cup \vec{\mu}_1^F \cup \ldots \cup \vec{x}_{f_F}^F \cup \vec{\mu}_{f_F}^F)).\end{aligned}$$

The semantics of γ is defined as follows. Let $\mathcal{A}$ be a structure and α be an assignment to $\vec{w}$ in $\mathcal{A}$. Let

$$\gamma^{(\mathcal{A},\alpha)} := \begin{cases} E(\mathcal{I}(\mathcal{A},\alpha)) & \text{if } \mathcal{I}(\mathcal{A},\alpha) \text{ is defined} \\ 0 & \text{otherwise.} \end{cases}$$

We also introduce the notion of a *Boolean-valued generalised operator*. The definition is almost identical except that the evaluation function is Boolean-valued rather than number-valued and an application of a Boolean-valued generalised operator defines a formula rather than a number term. The extension of a logic by a Boolean-valued generalised operator is defined similarly and if $\gamma(\vec{w})$ is an application of a Boolean-valued generalised operator the semantics of γ is defined such that $\mathcal{A} \models \gamma[\alpha]$ if, and only if, $E(\mathcal{I}(\mathcal{A},\alpha)) = 1$. We use the term *generalised operator* to refer to either a number-valued or Boolean-valued generalised operator.

We are often interested in families of generalised operators generated by a single evaluation function. Let E be an evaluation function. Let $\boldsymbol{\Omega}_E$ denote the set of all operators $\Omega_{E,\mathrm{ar}}$ for every function $\mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0$. We call $\boldsymbol{\Omega}_E$ a *vectorised family of generalised operators* (or just a *vectorised operator*) and say it is *generated* by E. When a generalised operator appears in a formula the arity of the operator can be deduced from the sequences of bound variables. As such, when an operator in $\boldsymbol{\Omega}_E$ appears in a formula or number term we often denote it by $\boldsymbol{\Omega}_E$ instead of writing $\Omega_{E,\mathrm{ar}}$ for some arity ar. The vectorised operator $\boldsymbol{\Omega}_E$ contains an operator of every possible arity. We are sometimes interested in those generalised operators that exclusively bind either element variables or number variables. Let $\boldsymbol{\Omega}_E$ be a vectorised operator. For each $i \in [2]$ let $\boldsymbol{\Omega}_E^i \subseteq \boldsymbol{\Omega}_E$ consist of all $\Omega_{E,\mathrm{ar}}$ such that for all

$j \in [2] \setminus \{i\}$, $s \in \boldsymbol{S}$ we have $\mathrm{ar}(s, j) = 0$. We call $\boldsymbol{\Omega}^1_E$ an *element-domain* vectorised family of generalised operators and $\boldsymbol{\Omega}^2_E$ a *number-domain* vectorised family of generalised operators.

Let $\boldsymbol{\Omega}$ be a set of generalised operators and ρ be a relational vocabulary. Let $L(\widetilde{\boldsymbol{\Omega}})[\rho]$ be the set of all formulas and number-terms in $L(\boldsymbol{\Omega})[\rho]$ such that each application of a generalised operator in $\boldsymbol{\Omega}$ that appears as a subformula or sub-number-term has trivial domain and equality formulas (i.e. the domain formulas are valid and equality formulas define equality between tuples). In other words, each application of a generalised operator defines an interpretation that involves no restriction of the domain and no quotienting. In this case, since the domain and equality formulas have no effect on the semantics, we omit them when applying the generalised operator.

Definition 3.4. Let $\boldsymbol{\Omega}$ be a set of operators. We say a logic $L(\boldsymbol{\Omega})$ is *closed under $\boldsymbol{\Omega}$-quotients* if for any finite relational vocabulary ρ we have $L(\widetilde{\boldsymbol{\Omega}})[\rho] \equiv L(\boldsymbol{\Omega})[\rho]$. If the set of operators $\boldsymbol{\Omega}$ is clear from context we say that $L(\boldsymbol{\Omega})$ is *closed under operator quotients*.

We now discuss the relationship between generalised operators and other means of extending logics. We first show that generalised operators generalise Lindström quantifiers. Let Q be a Lindström quantifier defined by a class of ρ-structures $\mathcal{C}$ for some relational vocabulary ρ. We think of ρ as a many-sorted vocabulary with a single sort. Let $E_{\mathcal{C}} : \mathrm{fin}^{\mathbb{N}}[\rho] \to \{0, 1\}$ be the characteristic function of $\mathcal{C}$ and let $\mathrm{ar}_Q : \{s\} \times [2] \to \mathbb{N}_0$ be defined such that $\mathrm{ar}_Q(s, 1) = 1$ and $\mathrm{ar}_Q(s, 2) = 0$. We identify Q with the Boolean-valued generalised operator $\Omega_{E_{\mathcal{C}},\mathrm{ar}_Q}$ and note that for any logic L we have $L(Q) \equiv L(\Omega_{E_{\mathcal{C}},\mathrm{ar}_Q})$. In particular it follows that the universal, existential, and counting quantifiers can all be thought of as generalised operators. Furthermore, a similar argument can be used to show that any vectorised family of quantifiers $\mathbf{Q} = (Q_n)_{n\in\mathbb{N}}$ generated by a class of structures $\mathcal{C}$ can be identified with an element-domain vectorised family of generalised operators $\boldsymbol{\Omega}^1_{E_{\mathcal{C}}}$. The 'element-domain' restriction is important as Lindström quantifiers can only bind element-sort variables. The counting operator can be understood as a number-valued generalised operator. Let $\tau_{\mathbf{set}} := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$, where $\boldsymbol{R} = \{U\}$, $\boldsymbol{F} = \emptyset$, and $\boldsymbol{S} = \{s\}$, and $\zeta(U) = (s)$. Let $E_{\mathbf{cnt}} : \mathrm{fin}[\tau_{\mathbf{set}}] \to \mathbb{N}_0$ be defined for each $\mathcal{A} \in \mathrm{fin}[\tau_{\mathbf{set}}]$ by $E_{\mathbf{cnt}}(\mathcal{A}) = |\mathcal{A}|$. We identify the counting operator $\#$ with the vectorised operator $\boldsymbol{\Omega}_{E_{\mathbf{cnt}}}$. It can be shown that $\mathrm{FPC} \equiv \mathrm{FP}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}}_{E_{\mathbf{cnt}}})$ and $\mathrm{FOC} \equiv \mathrm{FO}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}}_{E_{\mathbf{cnt}}})$. The rank operator can also be understood as a number-valued vectorised operator. Let $\tau_{\mathbf{rk}} := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$, where $\boldsymbol{R} = \emptyset$, $\boldsymbol{S} = [2]$, $F = \{M, p\}$, and $\zeta(M) = (1, 2)$ and $\zeta(p) = ()$. We think of a $\tau_{\mathbf{rk}}$-structure $\mathcal{B}$ as a number-valued matrix $M^{\mathcal{B}}$ together with a constant $p^{\mathcal{B}}$ intended to denote the characteristic of a finite field. Let $E_{\mathbf{rk}} : \mathrm{fin}^{\mathbb{N}}[\tau_{\mathbf{rk}}] \to \mathbb{N}_0$ be defined for each $\mathcal{B} \in \mathrm{fin}[\tau_{\mathbf{rk}}]$ such that

$$E_{\mathbf{rk}}(\mathcal{M}) = \begin{cases} \mathbf{rk}(M^{\mathcal{B}} \bmod p^{\mathcal{B}}) & \text{if } p^{\mathcal{B}} \text{ is prime} \\ 0 & \text{otherwise,} \end{cases}$$

where '$M^{\mathcal{B}}$ mod $p^{\mathcal{B}}$' is the matrix with entries in $\mathbb{F}_{p^{\mathcal{B}}}$ defined by taking the residue of each entry in the matrix $M^{\mathcal{B}}$ modulo $p^{\mathcal{B}}$. We identify the rank operator with the vectorised operator $\mathbf{\Omega}_{E_{\mathbf{rk}}}$. It can be shown that $\mathrm{FPR} \equiv \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{rk}}})$ and $\mathrm{FOR} \equiv \mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{rk}}})$.

It is known that FPC is closed under operator quotients. To see this it suffices to show that for any $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{cnt}}})$-definable congruence $\approx$ it is possible to count the number of $\approx$-equivalence-classes in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{cnt}}})$. This can be shown by first noting that it is possible to define a number term in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{cnt}}})$ that for a given natural number k (from a bounded set) denotes the cardinality of the set of all tuples contained in some $\approx$-equivalence-class with exactly k elements. We can then divide by k in order to define a number term that denotes the number of $\approx$-equivalence-classes of size k and use the fixed-point operator to sum over all relevant values of k and hence define a number term in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{cnt}}})$ that denotes the total number of $\approx$-equivalence-classes. For a complete proof please see [38]. We now show that FPR is also closed under operator quotients.

Lemma 3.5. $\mathrm{FPC} \equiv \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{cnt}}) \equiv \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}_{E_{cnt}})$ *and* $\mathrm{FPR} \equiv \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{rk}}) \equiv \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}_{E_{rk}})$.

Proof. Let θ be a $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}_{E_{\mathbf{rk}}})$-formula. We aim to prove by structural induction on θ that there exists some $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{rk}}})$-formula θ' that defines the same query as θ. The only interesting case is for an application of the rank operator. Let γ be a sub-number-term of θ of the form

$$\gamma \equiv \mathbf{\Omega}_{E_{\mathbf{rk}}}[(\phi^D_{s_1}, \phi^D_{s_2}), (\phi^{\approx}_{s_1}, \phi^{\approx}_{s_2})][\pi][(\vec{x}^R_1 \vec{\mu}^R_1, \vec{x}^R_2 \vec{\mu}^R_2)\, \nu],$$

where ϕ^D_1 and ϕ^D_2 are the domain formulas, $\phi^{\approx}_1$ and $\phi^{\approx}_2$ are the equality formulas, and ν and π are number terms. Let $\mathcal{I}_\gamma$ be the interpretation defined by these formulas and number terms. From the induction hypothesis there exist corresponding formulas and number terms ψ^D_1, ψ^D_2, $\psi^{\approx}_1$, $\psi^{\approx}_2$, ν', and π' definable in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}_{E_{\mathbf{rk}}})$. Let $\mathcal{I}'$ be the interpretation defined by these formulas and number terms. Let ψ^{chk} be the first-order formula with no free variables that checks whether $\mathcal{I}'$ is defined for a given structure and assignment (i.e. whether $\psi^{\approx}_1$, $\psi^{\approx}_2$ define a congruence on the domain). Let

$$\overline{\nu} :\equiv \#_\eta[(\eta = 1) \wedge \psi^{\mathrm{chk}}] \cdot \nu'$$

and let

$$\gamma' \equiv \mathbf{\Omega}_{E_{\mathbf{rk}}}[\pi'][(\vec{x}^R_1 \vec{\mu}^R_1, \vec{x}^R_2 \vec{\mu}^R_2)\, \overline{\nu}].$$

Let $\mathcal{I}_{\gamma'}$ be the interpretation defined in γ'. Let $\mathcal{A}$ be a structure with the same vocabulary as θ and let α be an assignment to the free variables in γ. Note that if $\mathcal{I}_\gamma(\mathcal{A}, \alpha)$ is not defined then $\mathcal{I}'(\mathcal{A}, \alpha)$ is not defined and so $\mathcal{A} \not\models \psi^{\mathrm{chk}}[\alpha]$ and $(\gamma')^{(\mathcal{A},\alpha)} = 0 = \gamma^{(\mathcal{A},\alpha)}$. Suppose $\mathcal{I}_\gamma(\mathcal{A}, \alpha)$ is defined. Let M_γ be the matrix defined by $\mathcal{I}_\gamma(\mathcal{A}, \alpha)$ and let $M_{\gamma'}$ be the matrix defined by $\mathcal{I}_{\gamma'}$. Note that since $\mathcal{I}_\gamma(\mathcal{A}, \alpha)$ and $\mathcal{I}'(\mathcal{A}, \alpha)$ are defined, the pairs of equality formulas in these

interpretations each define a congruence. It can be shown that $M_{\gamma'}$ is definable from M_γ by adding for each row (resp. column) in M_γ a copy of that row (resp. column) for each member of the equivalence class of that row (resp. column) index. Since rank is preserved under the addition of copies of rows or columns to the matrix it follows that $M_{\gamma'}$ and M_γ have the same rank, and hence $\gamma^{(\mathcal{A},\alpha)} = (\gamma')^{(\mathcal{A},\alpha)}$. The result follows by induction. □

It is often necessarily to restrict our attention to generalised operators defined by polynomially bounded evaluation functions. We define this property formally now.

Definition 3.6. Let $\Omega_{E,\mathrm{ar}}$ be a generalised operator. Let τ be the vocabulary of $\Omega_{E,\mathrm{ar}}$. We say that $\Omega_{E,\mathrm{ar}}$ is P-*bounded* if there is a polynomial p such that for any number-extended τ-structure $\mathcal{A}$ we have that $E(\mathcal{A}) < p(|\mathcal{A}|)$. We say a vectorised family of generalised operators is P-bounded if every generalised operator in the family is P-bounded.

We can also speak of number terms, formulas, and logics being P-bounded.

Definition 3.7. Let L be a logic and let ρ be a relational vocabulary. Let $\theta \in L[\rho]$. We say that θ is P-*bounded* if there is a polynomial p such that for any number-term γ appearing in θ, any ρ-structure $\mathcal{A}$, and any assignment α in $\mathcal{A}$ to the free variables of γ, it follows that $\gamma^{(\mathcal{A},\alpha)} < p(|\mathcal{A}|)$. We say a logic L is P-*bounded* if for any relational vocabulary ρ and any $\theta \in L[\rho]$ we have that θ is P-bounded.

It can be shown that if L is a P-bounded logic and $\mathbf{\Omega}$ is a family of P-bounded operators then $L(\mathbf{\Omega})$ is a P-bounded logic. An important property of P-bounded logics is that for any formula θ in $L(\mathbf{\Omega})$ there is some $k \in \mathbb{N}$ such that any number-term γ appearing in θ denotes a number that can be 'stored' in a k-tuple of number variables (in other words, if $\vec{\mu}$ is a k-sequence of number variables then '$\exists\vec{\mu}\,(\vec{\mu} = \gamma)$' holds for every finite structure).

It can be shown that every Boolean-valued generalised operator is P-bounded. The counting vectorised operator ($\mathbf{\Omega}_{E_{\mathbf{cnt}}}$) and rank vectorised operator ($\mathbf{\Omega}_{E_{\mathbf{rk}}}$) are also P-bounded. It is known that $\mathrm{FO}^{\mathbb{N}}$ and $\mathrm{FP}^{\mathbb{N}}$ are P-bounded. We can conclude the following.

Lemma 3.8. *The logics* $\mathrm{FO}^{\mathbb{N}}$, $\mathrm{FP}^{\mathbb{N}}$, FOC, FOR, FPC, *and* FPR *are* P-*bounded.*

We now introduce some useful terminology for generalised operators.

Definition 3.9. Let Ω be a generalised operator. Let τ and ar be the vocabulary and arity of Ω. We say Ω is *relational* if the vocabulary of τ does not contain any function symbols. We say Ω is *almost relational* if all the function symbols in τ are constants. We say Ω *has no constants* if τ does not contain any constant symbols. We say that Ω *quantifies over the universe* if $\mathrm{ar}(s, 2) = 0$ for every sort symbol in τ. We say that Ω *quantifies over the number-sort* if $\mathrm{ar}(s, 1) = 0$ for every sort symbol in τ. We say Ω is single-sorted if τ has a single sort symbol. For each property defined above we say that a vectorised family of operators has that property if every generalised operator in the family has that property.

We aim to define a translation for each extension of a fixed-point logic to either an infinitary logic (see Section 3.5) or to families of symmetric circuits (see Chapter 8). In order to define this translation we need to restrict our attention to logics extended by almost relational generalised operators. But this poses a problem as many logics of interest (e.g. FPR) would be excluded by this restriction. We now show that in many cases these logics are equivalent to a logic defined in terms of almost relational generalised operators. Before we formally state this result we first introduce a way of comparing the expressive power of two logics that takes into account both the queries expressible by formulas *and* the functions definable by number-terms.

Definition 3.10. Let L and L' be logics. We say $L \leq_{\mathbb{N}} L'$ if $L \leq L'$ and for any relational vocabulary ρ and any mixed sort sequence of variables $\vec{w}$ and any number-term $\gamma(\vec{w})$ definable in $L[\rho]$ there exists a number-term $\gamma'(\vec{w})$ definable in $L'[\rho]$ such that for any ρ-structure $\mathcal{A}$ and any assignment α to $\vec{w}$ in $\mathcal{A}$ we have that $\gamma^{(\mathcal{A},\alpha)} = (\gamma')^{(\mathcal{A},\alpha)}$. We write $L \equiv_{\mathbb{N}} L'$ if $L \leq_{\mathbb{N}} L'$ and $L' \leq_{\mathbb{N}} L$.

We note that if both L and L' do not have a number sort then $L \leq L'$ if, and only if, $L \leq_{\mathbb{N}} L'$ and if both do have a number sort then $L \leq_{\mathbb{N}} L'$ implies $L \leq L'$ and $L \equiv_{\mathbb{N}} L'$ implies $L \equiv L'$. If L is a logic with a number sort and $L \equiv_{\mathbb{N}} L(\mathbf{\Omega}_{E_{\mathbf{cnt}}})$ we say that L can *simulate counting*. We have that FPR can simulate counting. If L can simulate counting we often abuse notation and use counting operators when defining a number-term as any application of a counting operator can be replaced by an equivalent number-term definable in L.

We now show that if a logic L can simulate counting and $\mathbf{\Omega}$ is a P-bounded vectorised operator then we can define an equivalent P-bounded almost relational vectorised operator $\mathbf{\Omega}'$ such that $L(\mathbf{\Omega}) \equiv_{\mathbb{N}} L(\mathbf{\Omega}')$. We define $\mathbf{\Omega}'$ by adding a new sort and replacing each non-nullary function with a relation so that the value of the function for a given tuple is given by the number of elements in the new sort that relate to that tuple. In other words, we replace each non-constant function F with arity f with a $(f+1)$-ary relation R_F defined such that for each $\vec{a} \in \mathbf{Dom}(F)$ it follows that $F(\vec{a})$ is precisely equal to the number of elements c in the $(f+1)$th sort of R_F such that $(\vec{a}, c) \in R_F$. The proof of this result is mostly straightforward, but the introduction of a new sort does require a somewhat non-obvious construction in order to establish one direction of the equivalence. In order to illiterate the idea behind this construction we first consider the special case of the rank operator $\mathbf{\Omega}_{E_{\mathbf{rk}}}$ and show how we can construct a corresponding almost relational vectorised operator $\mathbf{\Omega}_{E_{\mathbf{rk}'}}$.

Example 3.11. We now explicitly define the P-bounded almost relational vectorised operator $\mathbf{\Omega}_{E'_{\mathbf{rk}}}$. We now define $E'_{\mathbf{rk}}$. Let $\tau'_{\mathbf{rk}} := (\{R\}, \{p\}, [3], \zeta)$ be defined such that $\zeta(R) = (1, 2, 3)$ and $\zeta(p) = ()$. Let $E'_{\mathbf{rk}} : \mathrm{fin}^{\mathbb{N}}[\tau'_{\mathbf{rk}}] \to \mathbb{N}_0$ be defined as follows. Let $\mathcal{B} \in \mathrm{fin}^{\mathbb{N}}[\tau'_{\mathbf{rk}}]$ and let $B_1 \uplus B_2 \uplus B_3$ be the universe of $\mathcal{B}$. If $p^{\mathcal{B}}$ is prime let $M_{\mathcal{B}} : B_1 \times B_2 \to \mathbb{F}_p$ be defined such

that $M_{\mathcal{B}}(b_1, b_2) = |\{b_3 \in B_3 : (b_1, b_2, b_3) \in R^{\mathcal{B}}\}| \mod p^{\mathcal{B}}$ for all $(b_1, b_2) \in B_1 \times B_2$. Then

$$E'_{\mathbf{rk}}(\mathcal{B}) = \begin{cases} \mathbf{rk}(M_{\mathcal{B}}) & \text{if } p^{\mathcal{B}} \text{ is a prime} \\ 0 & \text{otherwise.} \end{cases}$$

It can be shown that $\mathrm{FPR} \equiv \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}_{E'_{\mathbf{rk}}})$.

Proposition 3.12. *Let $\mathbf{\Omega}_E$ be a* P*-bounded vectorised operator. There exists an almost relational vectorised operator $\mathbf{\Omega}_{E'}$ such that for any* P*-bounded logic L with a number sort we have that*

1. *$L(\mathbf{\Omega}_E) \leq_{\mathbb{N}} L(\mathbf{\Omega}_{E'})$,*
2. *if $L(\mathbf{\Omega}_E)$ can simulate counting then $L(\mathbf{\Omega}_{E'}) \leq_{\mathbb{N}} L(\mathbf{\Omega}_E)$,*
3. *$L(\widetilde{\mathbf{\Omega}}_E) \leq_{\mathbb{N}} L(\widetilde{\mathbf{\Omega}}_{E'})$, and*
4. *if $L(\widetilde{\mathbf{\Omega}}_E) \equiv_{\mathbb{N}} L(\widetilde{\mathbf{\Omega}}_E)(\widetilde{\mathbf{\Omega}}_{E_{cnt}})$ then $L(\widetilde{\mathbf{\Omega}}_{E'}) \leq_{\mathbb{N}} L(\widetilde{\mathbf{\Omega}}_E)$.*

Proof. Let $\tau := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be the vocabulary of $\mathbf{\Omega}_E$. Let $\tau' := (\boldsymbol{R}', \boldsymbol{F}', \boldsymbol{S}', \zeta')$ where

- $\boldsymbol{R}' = R \uplus \{R_F : F \in \boldsymbol{F}, f_F \geq 1\}$,
- $\boldsymbol{F}' = \{F \in \boldsymbol{F} : f_F = 0\}$,
- $\boldsymbol{S}' = \boldsymbol{S} \uplus \{s_c\}$ (where s_c is some sort symbol),
- for all $F \in \boldsymbol{F}'$, $\zeta'(F) = \zeta(F) = ()$, and
- for all $R \in \boldsymbol{R}'$ if $R \in \boldsymbol{R}$ then $\zeta'(R) = \zeta(R)$ and otherwise there exists $F \in \boldsymbol{F}$ such that $R = R_F$ and $\zeta'(R) = (s_1, \ldots, s_{f_F}, s_c)$, where $(s_1, \ldots, s_{f_F}) = \zeta(F)$.

Let $T : \mathrm{fin}^{\mathbb{N}}[\tau'] \to \mathrm{fin}^{\mathbb{N}}[\tau]$ be defined as follows. Let $\mathcal{B}' \in \mathrm{fin}^{\mathbb{N}}[\tau']$ and let $B' := \uplus_{s \in \boldsymbol{S}'} B'_s$ be the universe of $\mathcal{B}'$. Let $T(\mathcal{B}')$ be the number extended τ-structure with universe $\uplus_{s \in \boldsymbol{S}} B'_s$ and such that for all $R \in \boldsymbol{R}$, $R^{T(\mathcal{B}')} = R^{\mathcal{B}'}$ and for all $F \in \boldsymbol{F}$ if F is a constant symbol then $F^{T(\mathcal{B}')} = F^{\mathcal{B}'}$ and otherwise $F^{T(\mathcal{B}')}(b_1, \ldots, b_{f_F}) = |\{b \in B'_{s_c} : (b_1, \ldots, b_{f_F}, b) \in R_F^{\mathcal{B}'}\}|$. Let $E' := E \circ T$.

Let L be a P-bounded logic with a number sort. Let ρ be a relational vocabulary. We now prove Statement 1. Let $\theta(\vec{x}) \in L(\mathbf{\Omega}_E)[\rho]$. Let $t \in \mathbb{N}$ be such that $p(n) = n^t$ is a polynomial witnessing the fact that θ is P-bounded. We aim to prove by induction on the structure of the formula that for each subformula (resp. sub-number-term) $\gamma(\vec{w})$ of $\theta(\vec{x})$ there is a corresponding formula (resp. number-term) $\gamma'(\vec{w})$ in $L(\mathbf{\Omega}_{E'})$ such that for any ρ-structure $\mathcal{A}$ and assignment α to $\vec{w}$ in $\mathcal{A}$ we have that $\gamma^{(\mathcal{A},\alpha)} = (\gamma')^{(\mathcal{A},\alpha)}$. We say that γ' is a *translation* of γ. The only non-trivial case in this induction is when γ is an application of a generalised operator. Suppose

$$\gamma(\vec{w}) \equiv \Omega_{E,\mathrm{ar}}[(\phi_s^D)_{s \in \boldsymbol{S}}, (\phi_s^{\approx})_{s \in \boldsymbol{S}}][(\vec{x}_1^R \vec{\mu}_1^R, \ldots, \vec{x}_{r_R}^R \vec{\mu}_{r_R}^R)\, \phi_R]_{R \in \boldsymbol{R}}[(\vec{y}_1^F \vec{\nu}_1^F, \ldots, \vec{y}_{f_F}^F \vec{\nu}_{f_F}^F)\, \eta_F]_{F \in \boldsymbol{F}},$$

where for each $s \in \boldsymbol{S}$ the free variables in ϕ_s^D are among $(\vec{x}^s\vec{\mu}^s, \vec{w})$, the free variables in $\phi_s^{\approx}$ are among $(\vec{x}_1^s\vec{\mu}_1^s, \vec{x}_2^s\vec{\mu}_2^s, \vec{w})$ and $\langle(\phi_s^D)_{s\in\boldsymbol{S}}, (\phi_s^{\approx})_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}}, (\eta_F)_{F\in\boldsymbol{F}}\rangle$ defines a number-extended $L(\boldsymbol{\Omega}_E)[\rho,\tau]$-interpretation with dimension ar and parameters $\vec{w}$. From the inductive hypothesis there is a translation for each subformula and sub-number-term of γ. Let $\mathrm{ar}' : \boldsymbol{S}' \times [2] \to \mathbb{N}_0$ be such that $\mathrm{ar}'(s,i) = \mathrm{ar}(s,i)$ for all $(s,i) \in \boldsymbol{S} \times [2]$, $\mathrm{ar}'(s_c, 1) = 0$, and $\mathrm{ar}'(s_c, 2) = t$. For each $s \in \boldsymbol{S}'$ if $s \in S$ let ψ_s^D be a translation of ϕ_s^D and $\psi_s^{\approx}$ be a translation of $\phi_s^{\approx}$ and otherwise let ψ_s^D be a valid formula and let $\psi_s^{\approx} \equiv (\vec{\mu}_1^s = \vec{\mu}_2^s)$. For each $F \in \boldsymbol{F}$ let χ_F be a translation of η_F. For each $R \in \boldsymbol{R}' \setminus \boldsymbol{R}$ and each $i \in [r_R]$ let $\vec{x}_i^R$ be a fresh $\mathrm{ar}'(\zeta'(R)(i), 1)$-sequence of element variables and let $\vec{\mu}_i^R$ be a fresh $\mathrm{ar}'(\zeta'(R)(i), 2)$-sequence of number variables. For each $R \in \boldsymbol{R}'$ if $R \in \boldsymbol{R}$ let ψ_R be a translation of ϕ_R and otherwise there exists $F \in \boldsymbol{F}$ such that $R = R_F$ and let

$$\psi_R((\vec{x}_1^R\vec{\mu}_1^R, \ldots, \vec{x}_{r_R}^R\vec{\mu}_{r_R}^R, \vec{w})) \equiv \vec{\mu}_{r_R}^R < \chi_F(\vec{x}_1^R\vec{\mu}_1^R, \ldots, \vec{x}_{r_R-1}^R\vec{\mu}_{r_R-1}^R, \vec{w})).$$

We have defined ψ_R using an abbreviation introduced in Remark 2.1. Let

$$\gamma'(\vec{w}) \equiv \Omega_{E',\mathrm{ar}'}[(\psi_s^D)_{s\in\boldsymbol{S}'}, (\psi_s^{\approx})_{s\in\boldsymbol{S}'}][(\vec{x}_1^R\vec{\mu}_1^R, \ldots, \vec{x}_{r_R}^R\vec{\mu}_{r_R}^R)\,\psi_R]_{R\in\boldsymbol{R}'}[(\vec{y}_1^F\vec{\nu}_1^F, \ldots, \vec{y}_{f_F}^F\vec{\nu}_{f_F}^F)\,\chi_F]_{F\in\boldsymbol{F}'}.$$

Note that if $\boldsymbol{\Omega}_E$ is a Boolean-valued vectorised operator then $\boldsymbol{\Omega}_{E'}$ is also Boolean-valued and both γ and γ' are formulas. Otherwise $\boldsymbol{\Omega}_{E'}$ is number-valued and both γ and γ' are number-terms. Let $\mathcal{A} \in \mathrm{fin}[\rho]$ and let α be an assignment to $\vec{w}$ in $\mathcal{A}$. Let n be the size of $\mathcal{A}$. Let $\mathcal{I} := \langle(\phi_s^D)_{s\in\boldsymbol{S}}, (\phi_s^{\approx})_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}}, (\eta_F)_{F\in\boldsymbol{F}}\rangle$ be the number-extended $L(\boldsymbol{\Omega}_E)[\rho,\tau]$-interpretation defined in γ and let $\mathcal{I}' := \langle(\psi_s^D)_{s\in\boldsymbol{S}'}, (\psi_s^{\approx})_{s\in\boldsymbol{S}'}, (\psi_R)_{R\in\boldsymbol{R}'}, (\chi_F)_{F\in\boldsymbol{F}'}\rangle$ be the number-extended $L(\boldsymbol{\Omega}_{E'})[\rho,\tau']$-interpretation defined in γ'.

Claim 3.12.1. *If $\mathcal{I}(\mathcal{A},\alpha)$ is defined then $\mathcal{I}'(\mathcal{A},\alpha)$ is defined and there exists $\mathcal{B}' \in \mathrm{fin}^{\mathbb{N}}[\tau']$ such that $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$ and $\mathcal{I}(\mathcal{A},\alpha) = T(\mathcal{B}')$.*

Proof. Suppose $\mathcal{I}(\mathcal{A},\alpha)$ is defined and let $\mathcal{B} = \mathcal{I}(\mathcal{A},\alpha)$. Let h be the witnessing coordinate map. Let $B = \uplus_{s\in\boldsymbol{S}} B_s$ be the universe of $\mathcal{B}$. Let $\mathcal{B}'$ be the number-extended τ'-structure such that

- $\mathcal{B}'$ has universe $B' = \uplus_{s\in\boldsymbol{S}'} B'_s$ where for all $s \in \boldsymbol{S}'$ if $s \in \boldsymbol{S}$ then $B'_s = B_s$ and otherwise $B'_s = [(n+1)^t - 1]_0$;
- for each $R \in \boldsymbol{R}'$ if $R \in \boldsymbol{R}$ then $R^{\mathcal{B}'} = R^{\mathcal{B}}$ and otherwise there exists $F \in \boldsymbol{F}$ such that $R = R_F$ and $R^{\mathcal{B}'} = \{(b_1, \ldots, b_{f_F}, b) \in \mathbf{Dom}(F^{\mathcal{B}}) \times B_{s_c} : b < F^{\mathcal{B}}(b_1, \ldots, b_{f_F})\}$; and
- for each $F \in \boldsymbol{F}'$ we have $F^{\mathcal{B}'} = F^{\mathcal{B}}$.

Let $h' : \uplus_{s\in\boldsymbol{S}'}(\psi_s^D)^{(\mathcal{A},\alpha)} \to \uplus_{s\in\boldsymbol{S}'} B'_s$ be defined such that for each $\vec{a}\vec{m} \in \uplus_{s\in\boldsymbol{S}'}(\psi_s^D)^{(\mathcal{A},\alpha)}$ if $\vec{a}\vec{m} \in B_s$ for some $s \in \boldsymbol{S}$ then $h'(\vec{a}\vec{m}) = h(\vec{a}\vec{m})$ and otherwise $h'(\vec{a}\vec{m})$ is an element of B'_{s_c} and $h'(\vec{a}\vec{m}) = \sum_{l\in[|\vec{m}|]} m_l \cdot (n+1)^{l-1}$. It can be shown that h' is surjective.

We aim to show that $\mathcal{B}' = \mathcal{I}'(\mathcal{A}, \alpha)$ with coordinate map h'. Let $s \in \boldsymbol{S}'$ and let $\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2 \in (\psi_s^D)^{(\mathcal{A},\alpha)}$. Suppose $s \in \boldsymbol{S}$. Then $h'(\vec{a}_1\vec{m}_1) = h'(\vec{a}_2\vec{m}_2)$ if, and only if, $h(\vec{a}_1\vec{m}_1) = h(\vec{a}_2\vec{m}_2)$ if, and only if, $(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\phi_s^{\approx})^{(\mathcal{A},\alpha)}$ if, and only if, $(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\psi_s^{\approx})^{(\mathcal{A},\alpha)}$. Otherwise $s = s_c$ and $|\vec{a}_1| = |\vec{a}_2| = 0$ and, since $\psi_s^{\approx}$ defines equality and h' is injective on $(\psi_{s_c}^D)^{(\mathcal{A},\alpha)}$, it follows that $(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\psi_s^{\approx})^{(\mathcal{A},\alpha)}$ if, and only if, $h'(\vec{m}_1) = h'(\vec{m}_2)$. Let $R \in \boldsymbol{R}'$ and for each $i \in [r_R]$ let $\vec{a}_i\vec{m}_i \in (\psi^D_{\zeta'(R)(i)})^{(\mathcal{A},\alpha)}$. Suppose $R \in \boldsymbol{R}$. Then $R^{\mathcal{B}} = R^{\mathcal{B}'}$ and so $(h'(\vec{a}_1\vec{m}_1), \ldots, h'(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}'}$ if, and only if, $(h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}}$ if, and only if, $(\vec{a}_1\vec{m}_1, \ldots, \vec{a}_{r_R}, \vec{m}_{r_R}) \in \phi_R^{(\mathcal{A},\alpha)}$ if, and only if, $(\vec{a}_1\vec{m}_1, \ldots, \vec{a}_{r_R}, \vec{m}_{r_R}) \in \psi_R^{(\mathcal{A},\alpha)}$. Otherwise $R \notin \boldsymbol{R}$ and so $R = R_F$ for some $F \in \boldsymbol{F}$ and

$$
\begin{aligned}
(h'(\vec{a}_1\vec{m}_1), \ldots, h'(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}'} & \\
&\iff (h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{f_F}\vec{m}_{f_F}), h'(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}'} \\
&\iff h'(\vec{a}_{r_R}\vec{m}_{r_R}) < F^{\mathcal{B}}(h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{f_F}\vec{m}_{f_F})) \\
&\iff \sum_{l \in [|\vec{m}_{r_R}|]} \vec{m}_{r_R}(l) \cdot (n+1)^{l-1} < F^{\mathcal{B}}(h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{f_F}\vec{m}_{f_F})) \\
&\iff (\vec{a}_1\vec{m}_1, \ldots, \vec{a}_{r_R}\vec{m}_{r_R}) \in \psi_R^{(\mathcal{A},\alpha)}.
\end{aligned}
$$

For all $F \in \boldsymbol{F}'$, $F^{(\mathcal{B}',\alpha)} = F^{(\mathcal{B},\alpha)} = \eta^{(\mathcal{A},\alpha)} = \chi^{(\mathcal{A},\alpha)}$. It follows that $\mathcal{B}' = \mathcal{I}'(\mathcal{A}, \alpha)$ with coordinate map h'. We have $\mathcal{B} = T(\mathcal{B}')$ as

- $\uplus_{s \in \boldsymbol{S}} B_s$ is the universe of both $\mathcal{B}$ and $T(\mathcal{B}')$;
- for each $R \in \boldsymbol{R}$, $R^{\mathcal{B}} = R^{\mathcal{B}'} = R^{T(\mathcal{B}')}$;
- for each $F \in \boldsymbol{F}'$, $F^{\mathcal{B}} = F^{\mathcal{B}'} = F^{T(\mathcal{B}')}$; and
- for each $F \in \boldsymbol{F} \setminus \boldsymbol{F}'$ we have for all $(b_1, \ldots, b_{f_F}) \in \mathbf{Dom}(F^{\mathcal{B}})$ and $b \in B'_{s_c}$ that $b < F^{\mathcal{B}}(b_1, \ldots, b_{f_F})$ if, and only if, $(b_1, \ldots, b_{f_F}, b) \in R_F^{\mathcal{B}'}$, and so, since B'_{s_c} is an initial segment of $\mathbb{N}_0$ and $F^{\mathcal{B}}(b_1, \ldots, b_{f_F}) \in B'_{s_c}$, it follows that $F^{\mathcal{B}}(b_1, \ldots, b_{f_F}) = |\{b \in B'_{s_c} : b < F^{\mathcal{B}}(b_1, \ldots, b_{f_F})\}| = |\{(b \in B'_{s_c} : (b_1, \ldots, b_{f_F}, b) \in R_F^{\mathcal{B}'}\}| = F^{T(\mathcal{B}')}(b_1, \ldots, b_{f_F})$.

This completes the proof of Claim 3.12.1.

□

Claim 3.12.2. *If $\mathcal{I}'(\mathcal{A}, \alpha)$ is defined then $\mathcal{I}(\mathcal{A}, \alpha)$ is defined.*

Proof. Suppose $\mathcal{I}'(\mathcal{A}, \alpha)$ is defined and let $\mathcal{B}' := \mathcal{I}'(\mathcal{A}, \alpha)$. Let h' be the witnessing coordinate map. Let $B' = \uplus_{s \in \boldsymbol{S}'} B'_s$ be the universe of $\mathcal{B}'$. Let $\mathcal{B} := T(B')$ and let $B = \uplus_{s \in \boldsymbol{S}} B_s$ be the universe of $\mathcal{B}$. Note that for all $s \in \boldsymbol{S}$, $(\phi_s^D)^{(\mathcal{A},\alpha)} = (\psi_s^D)^{(\mathcal{A},\alpha)}$ and $B_s = B'_s$. Let h be the restriction of h' to $\uplus_{s \in \boldsymbol{S}} (\phi_s^D)^{(\mathcal{A},\alpha)}$. It remains to check that $\mathcal{B} = \mathcal{I}(\mathcal{A}, \alpha)$ with coordinate map h. We omit the details here. The argument is similar to the one used in Claim 3.12.1.

□

Suppose $\mathcal{I}(\mathcal{A},\alpha)$ is not defined. It follows from Claim 3.12.2 that $\mathcal{I}'(\mathcal{A},\alpha)$ is not defined. If $\mathbf{\Omega}_E$ is number-valued then $\gamma^{(\mathcal{A},\alpha)} = 0 = (\gamma')^{(\mathcal{A},\alpha)}$. Otherwise $\mathbf{\Omega}_E$ is Boolean-valued and $\mathcal{A} \not\models \gamma[\alpha]$ and $\mathcal{A} \not\models \gamma'[\alpha]$. Suppose instead that $\mathcal{I}(\mathcal{A},\alpha)$ is defined. It follows from Claim 3.12.1 that $\mathcal{I}'(\mathcal{A},\alpha)$ is defined and there exists a number-extended τ'-structure $\mathcal{B}'$ such that $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$ and $T(\mathcal{B}') = \mathcal{I}(\mathcal{A},\alpha)$. If $\mathbf{\Omega}_E$ is number-valued then $(\gamma')^{(\mathcal{A},\alpha)} = E'(\mathcal{B}') = E(T(\mathcal{B}')) = \gamma^{(\mathcal{A},\alpha)}$. Otherwise $\mathbf{\Omega}_E$ is Boolean-valued and $\mathcal{A} \models \gamma'[\alpha]$ if, and only if, $E'(\mathcal{B}') = 1$ if, and only if, $E(T(\mathcal{B}')) = 1$ if, and only if, $\mathcal{A} \models \gamma[\alpha]$. It follows by induction that each subformula and sub-number-term of $\theta(\vec{x})$ has a translation in $L(\mathbf{\Omega}_{E'})$ and, in particular, there exists $\theta'(\vec{x})$ in $L(\mathbf{\Omega}_{E'})$ such that $\theta'(\vec{x})$ is a translation of $\theta(\vec{x})$. Each number term $\gamma(\vec{w})$ definable in $L(\mathbf{\Omega}_E)$ appears in some formula $\theta(\vec{x}) \in L(\mathbf{\Omega}_E)$. This concludes the proof of Statement 1.

We now prove Statement 2. Suppose $L(\mathbf{\Omega}_E)$ can simulate counting. Let $\theta'(\vec{x})$ be a formula in $L(\mathbf{\Omega}_{E'})$. Let $t \in \mathbb{N}$ be such that $p(n) = n^t$ is a polynomial witnessing the fact that $\theta'(\vec{x})$ is P-bounded. We use an argument similar to that for Statement 1. We aim to prove by induction on the structure of $\theta(\vec{x})$ that for each subformula (resp. sub-number-term) $\gamma'(\vec{w})$ of $\theta'(\vec{w})$ there is a formula (resp. number-term) $\gamma(\vec{w})$ in $L(\mathbf{\Omega}_E)$ such that for any ρ-structure $\mathcal{A}$ and assignment α to $\vec{w}$ in $\mathcal{A}$ it follows that $\gamma^{(\mathcal{A},\alpha)} = (\gamma')^{(\mathcal{A},\alpha)}$. We say that γ is a *translation* of γ'. Again, the only interesting case is the application of an operator. Suppose

$$\gamma'(\vec{w}) \equiv \Omega_{E',\mathrm{ar}'}[(\psi^D_s)_{s\in\boldsymbol{S}'}, (\psi^{\approx}_s)_{s\in\boldsymbol{S}'}][(\vec{x}^R_1\vec{\mu}^R_1, \ldots, \vec{x}^R_{r_R}\vec{\mu}^R_{r_R})\,\psi_R]_{R\in\boldsymbol{R}'}[(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F})\,\chi_F]_{F\in\boldsymbol{F}'}$$

where for each $s \in \boldsymbol{S}$ the free variables in ψ^D_s are among $(\vec{x}^s\vec{\mu}^s, \vec{w})$, the free variables in $\psi^{\approx}_s$ are among $(\vec{x}^s_1\vec{\mu}^s_1, \vec{x}^s_2\vec{\mu}^s_2, \vec{w})$ and $\langle (\psi^D_s)_{s\in\boldsymbol{S}}, (\psi^{\approx}_s)_{s\in\boldsymbol{S}}, (\psi_R)_{R\in\boldsymbol{R}}, (\chi_F)_{F\in\boldsymbol{F}}\rangle$ defines a number-extended $L(\mathbf{\Omega}_{E'})[\rho,\tau]$-interpretation with dimension ar' and parameters $\vec{w}$. From the inductive hypothesis there is a translation for each subformula and sub-number-term of $\gamma'(\vec{w})$. We aim to define a formula or number term $\gamma(\vec{w})$ in $L(\mathbf{\Omega}_E)$ (as appropriate) such that $\gamma(\vec{w})$ is a translation of $\gamma'(\vec{w})$. There is a small complication that should be discussed. A natural approach might involve translating the domain and equality formulas in γ' and letting γ be an application of $\Omega_{E,\mathrm{ar}'}$ with these translations as domain and equality formulas. However, the vocabulary of the operator $\Omega_{E',\mathrm{ar}'}$ has one additional symbol s_c not in the vocabulary of $\Omega_{E,\mathrm{ar}'}$. We cannot just ignore the domain and equality formulas for s_c as they may determine if the interpretation in γ' is defined. As such, when we define γ we include a gadget to check if the equality formula for s_c defines an equivalence relation and if it extends to a congruence. We now proceed with the translation. For each $s \in \boldsymbol{S}'$ let ϕ^D_s be a translation of ψ^D_s and let $\phi^{\approx}_s$ be a translation of $\psi^{\approx}_s$. For each $R \in \boldsymbol{R}'$ let ϕ_R be a translation of ψ_R and for each $F \in \boldsymbol{F}'$ let η_F be a translation of χ_F. Let $\mathrm{ar} = \mathrm{ar}'\big|_{\boldsymbol{S}\times[2]}$. For each $F \in \boldsymbol{F} \setminus \boldsymbol{F}'$ and each $i \in [f_F]$ let $\vec{y}^F_i$ be an $\mathrm{ar}(\zeta(F)(i),1)$-sequence of element variables and let $\vec{\nu}^F_i$ be an $\mathrm{ar}(\zeta(F)(i),2)$-sequence of number variables. For each $F \in \boldsymbol{F} \setminus \boldsymbol{F}'$ let $R := R_F$ and let

$$\eta_F(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F}, \vec{w}) \equiv \mathbf{\Omega}_{E_{\mathbf{cnt}}}[\phi^D_{s_c}, \phi^{\approx}_{s_c}][(\vec{x}^R_{r_R}\vec{\mu}^R_{r_R})\,\phi_R(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F}, \vec{\mu}^R_{r_R}\vec{\mu}^R_{r_R}, \vec{w})].$$

We now define the gadgets used to check if $\phi^{\approx}_{s_c}$ defines an equivalence relation and if the equivalence relation defined in γ' is a congruence. Let

$$\begin{aligned}\theta_{s_c}(\vec{w}) \equiv \forall \vec{x}_1\vec{\mu}_1, \vec{x}_2\vec{\mu}_2, \vec{x}_3\vec{\mu}_3\ [&(\phi^D_{s_c}(\vec{x}_1\vec{\mu}_1, \vec{w}) \wedge \phi^D_{s_c}(\vec{x}_2\vec{\mu}_2, \vec{w}) \wedge \phi^D_{s_c}(\vec{x}_3\vec{\mu}_3, \vec{w})) \implies \\ &[(\phi^{\approx}_{s_c}(\vec{x}_1\vec{\mu}_1, \vec{x}_2\vec{\mu}_2, \vec{w}) \iff \phi^{\approx}_{s_c}(\vec{x}_2\vec{\mu}_2, \vec{x}_1\vec{\mu}_1, \vec{w}))\wedge \\ &(\phi^{\approx}_{s_c}(\vec{x}_1\vec{\mu}_1, \vec{x}_2\vec{\mu}_2, \vec{w}) \wedge \phi^{\approx}_{s_c}(\vec{x}_2\vec{\mu}_2, \vec{x}_3\vec{\mu}_3, \vec{w}) \implies \phi^{\approx}_{s_c}(\vec{x}_1\vec{\mu}_1, \vec{x}_3\vec{\mu}_3, \vec{w}))\wedge \\ &\phi^{\approx}_{s_c}(\vec{x}_1\vec{\mu}_1, \vec{x}_1\vec{\mu}_1, \vec{w})]]\end{aligned}$$

and for each $R \in \boldsymbol{R}' \setminus \boldsymbol{R}$ let

$$\begin{aligned}\theta_R(\vec{w}) \equiv\ \forall \vec{x}_1\vec{\mu}_1, \ldots, \vec{x}_{r_R}\vec{\mu}_{r_R}, \vec{x}'_1\vec{\mu}'_1, \ldots, \vec{x}'_{r_R}\vec{\mu}'_{r_R}\ [&(\bigwedge_{i\in[r_R]} \phi^D_{\zeta'(R)(i)}(\vec{x}_i\vec{\mu}_i, \vec{w}) \wedge \phi^D_{\zeta'(R)(i)}(\vec{x}'_i\vec{\mu}'_i, \vec{w})) \implies \\ &[(\bigwedge_{i\in[r_R]} \phi^{\approx}_{\zeta'(R)(i)}(\vec{x}_i\vec{\mu}_i, \vec{x}'_i\vec{\mu}'_i, \vec{w})) \implies \\ &(\phi_R(\vec{x}_1\vec{\mu}_1, \ldots, \vec{x}_{r_R}\vec{\mu}_{r_R}, \vec{w}) \iff \phi_R(\vec{x}'_1\vec{\mu}'_1, \ldots, \vec{x}'_{r_R}\vec{\mu}'_{r_R}, \vec{w})]].\end{aligned}$$

Let $\theta_c \equiv \theta_{s_c} \wedge (\bigwedge_{R\in\boldsymbol{R}'\setminus\boldsymbol{R}} \theta_R)$. Let $\gamma_c \equiv \boldsymbol{\Omega}_{E_{\mathbf{cnt}}}[(\mu)\,(\theta_c \wedge \mu < 1)]$. If $\boldsymbol{\Omega}_E$ is Boolean-valued let

$$\gamma(\vec{w}) \equiv \theta_c \wedge \Omega_{E,\mathrm{ar}}[(\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\approx}_s)_{s\in\boldsymbol{S}}][(\vec{x}^R_1\vec{\mu}^R_1, \ldots, \vec{x}^R_{r_R}\vec{\mu}^R_{r_R})\,\phi_R]_{R\in\boldsymbol{R}}[(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F})\,\eta_F]_{F\in\boldsymbol{F}}$$

and if $\boldsymbol{\Omega}_E$ is number-valued let

$$\gamma(\vec{w}) \equiv \gamma_c \cdot \Omega_{E,\mathrm{ar}}[(\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\approx}_s)_{s\in\boldsymbol{S}}][(\vec{x}^R_1\vec{\mu}^R_1, \ldots, \vec{x}^R_{r_R}\vec{\mu}^R_{r_R})\,\phi_R]_{R\in\boldsymbol{R}}[(\vec{y}^F_1\vec{\nu}^F_1, \ldots, \vec{y}^F_{f_F}\vec{\nu}^F_{f_F})\,\eta_F]_{F\in\boldsymbol{F}}.$$

It remains to show that γ is a translation of γ'. Let $\mathcal{A} \in \mathrm{fin}[\rho]$ and let α be an assignment to $\vec{w}$ in $\mathcal{A}$. Let n be the size of $\mathcal{A}$. Let $\mathcal{I} = \langle (\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\approx}_s)_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}}, (\eta_F)_{F\in\boldsymbol{F}}\rangle$ be the number-extended $L(\boldsymbol{\Omega}_E)[\rho,\tau]$-interpretation defined in γ and let $\mathcal{I}' = \langle (\psi^D_s)_{s\in\boldsymbol{S}'}, (\psi^{\approx}_s)_{s\in\boldsymbol{S}'}, (\psi_R)_{R\in\boldsymbol{R}'}, (\chi_F)_{F\in\boldsymbol{F}'}\rangle$ be the number-extended $L(\boldsymbol{\Omega}_{E'})[\rho,\tau']$-interpretation defined in γ'.

Claim 3.12.3. *If $\mathcal{I}'(\mathcal{A},\alpha)$ is defined then $\mathcal{I}(\mathcal{A},\alpha)$ is defined and for any number-extended τ'-structure $\mathcal{B}'$ such that $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$ it follows that $T(\mathcal{B}') = \mathcal{I}(\mathcal{A},\alpha)$.*

Proof. Suppose $\mathcal{I}'(\mathcal{A},\alpha)$ is defined. Let $\mathcal{B}'$ be a number-extended τ'-structure such that $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$ and let h' be the witnessing coordinate map. Let $\uplus_{s\in\boldsymbol{S}'} B'_s$ be the universe of $\mathcal{B}'$. Let $\mathcal{B} := T(\mathcal{B}')$ and let $B = \uplus_{s\in\boldsymbol{S}} B_s$ be the universe of $\mathcal{B}$. Note that for all $s \in \boldsymbol{S}$ we have $B_s = B'_s$ and $(\psi^D_s)^{(\mathcal{A},\alpha)} = (\phi^D_s)^{(\mathcal{A},\alpha)}$. Let $h : \uplus_{s\in S}(\psi^D_s)^{(\mathcal{A},\alpha)} \to \uplus_{s\in\boldsymbol{S}} B_s$ be defined such that $h(\vec{a}\vec{m}) = h'(\vec{a}\vec{m})$ for all $\vec{a}\vec{m} \in \uplus_{s\in S}(\psi^D_s)^{(\mathcal{A},\alpha)}$. It remains to check that $\mathcal{B} = \mathcal{I}(\mathcal{A},\alpha)$ with coordinate map h. Since h' is surjective it follows that h is surjective. For each $s \in \boldsymbol{S}$ and all $\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2 \in (\phi^D_s)^{(\mathcal{A},\alpha)}$ we have $h(\vec{a}_1\vec{m}_1) = h(\vec{a}_2\vec{m}_2)$ if, and only if, $h'(\vec{a}_1\vec{m}_1) = h'(\vec{a}_2\vec{m}_2)$ if, and only if, $(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\psi^{\approx}_s)^{(\mathcal{A},\alpha)}$ if, and only if, $(\vec{a}_1\vec{m}_1, \vec{a}_2\vec{m}_2) \in (\phi^{\approx}_s)^{(\mathcal{A},\alpha)}$. Let $R \in \boldsymbol{R}$ and for each $i \in [r_R]$ let $\vec{a}_i\vec{m}_i \in (\phi^D_{\zeta(R)(i)})^{(\mathcal{A},\alpha)}$. Then $(h(\vec{a}_1\vec{m}_1), \ldots, h(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}}$ if,

and only if, $(h'(\vec{a}_1\vec{m}_1),\ldots,h'(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}'}$ if, and only if, $(\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{r_R}\vec{m}_{r_R}) \in \psi_R^{(\mathcal{A},\alpha)}$ if, and only if, $(\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{r_R}\vec{m}_{r_R}) \in \phi_R^{(\mathcal{A},\alpha)}$. Let $F \in \boldsymbol{F} \setminus \boldsymbol{F'}$ and for each i $\in [f_F]$ let $\vec{a}_i\vec{m}_i \in (\phi^D_{\zeta(F)(i)})^{(\mathcal{A},\alpha)}$. Then

$$\begin{aligned}
F^{\mathcal{B}}(h(\vec{a}_1\vec{m}_1),\ldots,h(\vec{a}_{f_F}\vec{m}_{f_F})) &= |\{b \in B'_{s_c} : (h(\vec{a}_1\vec{m}_1),\ldots,h(\vec{a}_{f_F}\vec{m}_{f_F}),b) \in R_F^{\mathcal{B}'}\}| \\
&= |\{h'(\vec{a}\vec{m}) : \vec{a}\vec{m} \in (\psi^D_{s_c})^{(\mathcal{A},\alpha)} \text{ and } (h'(\vec{a}_1\vec{m}_1),\ldots,h'(\vec{a}_{f_F}\vec{m}_{f_F}),h'(\vec{a}\vec{m})) \in R_F^{\mathcal{B}'}\} \\
&= |\{h'(\vec{a}\vec{m}) : \vec{a}\vec{m} \in (\psi^D_{s_c})^{(\mathcal{A},\alpha)} \text{ and } (\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{f_F}\vec{m}_{f_F},\vec{a}\vec{m}) \in \psi_{R_F}^{(\mathcal{A},\alpha)}\}| \\
&= |\{h'(\vec{a}\vec{m}) : \vec{a}\vec{m} \in (\phi^D_{s_c})^{(\mathcal{A},\alpha)} \text{ and } (\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{f_F}\vec{m}_{f_F},\vec{a}\vec{m}) \in (\phi_{R_F})^{(\mathcal{A},\alpha)}\}| \\
&= |\{\vec{a}\vec{m} \in (\phi^D_{s_c})^{(\mathcal{A},\alpha)} : (\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{f_F}\vec{m}_{f_F},\vec{a}\vec{m}) \in (\phi_{R_F})^{(\mathcal{A},\alpha)}\}/(\phi^{\approx}_{s_c})^{(\mathcal{A},\alpha)}| \\
&= \eta_F^{(\mathcal{A},\beta)} \text{ where } \beta = \alpha\frac{\vec{a}_1}{\vec{y}_1}\frac{\vec{m}_1}{\vec{\nu}_1}\ldots\frac{\vec{a}_{f_F}}{\vec{y}_{f_F}}\frac{\vec{m}_{f_F}}{\vec{\nu}_{f_F}}
\end{aligned}$$

We note that for all $F \in \boldsymbol{F'}$, F is a constant symbol and so $F^{\mathcal{B}} = \chi^{(\mathcal{A},\alpha)} = \eta^{(\mathcal{A},\alpha)}$. It follows that $T(\mathcal{B}') = \mathcal{B} = \mathcal{I}(\mathcal{A},\alpha)$ with coordinate map h. This completes the proof of Claim 3.12.3. □

Claim 3.12.4. *Suppose $\mathcal{I}'(\mathcal{A},\alpha)$ is not defined. If $\boldsymbol{\Omega}_E$ is number-valued then $\gamma^{(\mathcal{A},\alpha)} = 0$ and if $\boldsymbol{\Omega}_E$ is Boolean-valued then $\mathcal{A} \not\models \gamma[\alpha]$.*

Proof. Let us first suppose that $\boldsymbol{\Omega}_E$ is number-valued. We prove the contrapositive. Suppose $\gamma^{(\mathcal{A},\alpha)} \neq 0$. Then $\mathcal{I}(\mathcal{A},\alpha)$ is defined and $\gamma_c^{(\mathcal{A},\alpha)} > 0$. Let $\mathcal{B}$ be a number-extended τ-structure such that $\mathcal{B} = \mathcal{I}(\mathcal{A},\alpha)$ and let h be the witnessing coordinate map. It follows from $\gamma_c^{(\mathcal{A},\alpha)} > 0$ that $\mathcal{A} \models \theta_c[\alpha]$ and so, from the definition of θ_c, the binary relation defined by $(\psi^{\approx}_{s_c})^{(\mathcal{A},\alpha)}$ is an equivalence relation on $(\psi^D_{s_c})^{(\mathcal{A},\alpha)}$. We also have for all $s \in \boldsymbol{S}$ that $(\psi^D_s)^{(\mathcal{A},\alpha)} = (\phi^D_s)^{(\mathcal{A},\alpha)}$ and $(\psi^{\approx}_s)^{(\mathcal{A},\alpha)} = (\phi^{\approx}_s)^{(\mathcal{A},\alpha)}$ and so, since $(\phi^{\approx}_s)^{(\mathcal{A},\alpha)}$ defines an equivalence relation on $(\phi^D_s)^{(\mathcal{A},\alpha)}$, it follows that $(\psi^{\approx}_s)^{(\mathcal{A},\alpha)}$ defines an equivalence relation on $(\psi^D_s)^{(\mathcal{A},\alpha)}$.

Let $R \in \boldsymbol{R'}$. For each $i \in [r_R]$ let $\vec{a}_i\vec{m}_i, \vec{a}'_i\vec{m}'_i \in (\psi^D_s)^{(\mathcal{A},\alpha)}$ and suppose $(\vec{a}_i\vec{m}_i, \vec{a}'_i\vec{m}'_i) \in (\psi^{\approx}_{\zeta'(R)(i)})^{(\mathcal{A},\alpha)}$. If $R \in \boldsymbol{R}$ then

$$\begin{aligned}
(\vec{a}_1\vec{m}_1,\ldots,\vec{a}_{r_R}\vec{m}_{r_R}) \in (\psi_R)^{(\mathcal{A},\alpha)} &\iff (\vec{a}_1\vec{m}_2,\ldots,\vec{a}_{r_R}\vec{m}_{r_R}) \in (\phi_R)^{(\mathcal{A},\alpha)} \\
&\iff (h(\vec{a}_1\vec{m}_1),\ldots,h(\vec{a}_{r_R}\vec{m}_{r_R})) \in R^{\mathcal{B}} \\
&\iff (h(\vec{a}'_1\vec{m}'_1),\ldots,h(\vec{a}'_{r_R}\vec{m}'_{r_R})) \in R^{\mathcal{B}} \\
&\iff (\vec{a}'_1\vec{m}'_1,\ldots,\vec{a}'_{r_R}\vec{m}'_{r_R}) \in (\phi_R)^{(\mathcal{A},\alpha)} \\
&\iff (\vec{a}'_1\vec{m}'_1,\ldots,\vec{a}'_{r_R}\vec{m}'_{r_R}) \in (\psi_R)^{(\mathcal{A},\alpha)},
\end{aligned}$$

and otherwise $R \in \boldsymbol{R'} \setminus \boldsymbol{R}$ and so, since $\mathcal{A} \models \theta_c[\alpha]$ and hence $\mathcal{A} \models \theta_R[\alpha]$, it follows that $(\vec{a}_1\vec{m}_2,\ldots,\vec{a}_{r_R}\vec{m}_{r_R}) \in (\psi_R)^{(\mathcal{A},\alpha)}$ if, and only if, $(\vec{a}'_1\vec{m}'_2,\ldots,\vec{a}'_{r_R}\vec{m}'_{r_R}) \in (\psi_R)^{(\mathcal{A},\alpha)}$. Let $\mathcal{B}'$ be

the quotient of the number-extended τ'-structure

$$(\uplus_{s\in\boldsymbol{S}'}(\psi_s^D)^{(\mathcal{A},\alpha)}, (\psi_R^{(\mathcal{A},\alpha)})_{R\in\boldsymbol{R}'}, (\chi_F^{(\mathcal{A},\alpha)})_{F\in\boldsymbol{F}'})$$

by the equivalence relation $\uplus_{s\in\boldsymbol{S}'}(\psi_s^{\approx})^{(\mathcal{A},\alpha)}$. It follows that $\mathcal{I}'(\mathcal{A},\alpha)$ is defined and $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$. We can prove the claim for the case where $\mathbf{\Omega}_E$ is Boolean-valued using an essentially similar argument. This completes the proof of Claim 3.12.4.

□

Suppose $\mathcal{I}'(\mathcal{A},\alpha)$ is not defined. From Claim 3.12.4 it follows that if $\mathbf{\Omega}_E$ is number-valued then $(\gamma')^{(\mathcal{A},\alpha)} = 0 = \gamma^{(\mathcal{A},\alpha)}$ and if $\mathbf{\Omega}_E$ is Boolean-valued then $\mathcal{A} \not\models \gamma[\alpha]$ and $\mathcal{A} \not\models \gamma'[\alpha]$. Otherwise $\mathcal{I}'(\mathcal{A},\alpha)$ is defined and it follows from Claim 3.12.3 that $\mathcal{I}(\mathcal{A},\alpha)$ is defined and there exists a number-extended τ'-structure $\mathcal{B}'$ such that $T(\mathcal{B}') = \mathcal{I}(\mathcal{A},\alpha)$ and $\mathcal{B}' = \mathcal{I}'(\mathcal{A},\alpha)$. It follows from the fact that $\mathcal{I}'(\mathcal{A},\alpha)$ is defined that $\mathcal{A} \models \theta_c[\alpha]$ and so $(\gamma_c)^{(\mathcal{A},\alpha)} = 1$. If $\mathbf{\Omega}_E$ is number-valued we have that $\gamma^{(\mathcal{A},\alpha)} = (\gamma_c)^{(\mathcal{A},\alpha)} \cdot E(T(\mathcal{B}')) = E'(\mathcal{B}') = (\gamma')^{(\mathcal{A},\alpha)}$ and if $\mathbf{\Omega}_E$ is Boolean-valued then $\mathcal{A} \models \gamma[\alpha]$ if, and only if, $E(T(\mathcal{B}')) = 1$ if, and only if, $E'(\mathcal{B}') = 1$ if, and only if, $\mathcal{A} \models \gamma'[\alpha]$. It follows by induction that each subformula and sub-number-term of $\theta'(\vec{x})$ has a translation in $L(\mathbf{\Omega}_E)[\rho]$ and, in particular, there exists $\theta(\vec{x}) \in L(\mathbf{\Omega}_E)[\rho]$ such that $\theta(\vec{x})$ is a translation of $\theta'(\vec{x})$. We note that each number-term $\gamma'(\vec{w})$ definable in $L(\mathbf{\Omega}_{E'})[\rho]$ appears in some formula $\theta'(\vec{x}) \in L(\mathbf{\Omega}_{E'})[\rho]$. This concludes the proof of Statement 2.

In order to prove Statement 3 it suffices to note that in the translation in the proof of Statement 1 if $\gamma(\vec{w})$ has trivial domain and equality formulas (i.e. for each $s \in \boldsymbol{S}$ then ϕ_s^D is valid and $\phi^{\approx}$ defines equality) then $\gamma'(\vec{w})$ has trivial domain and equality formulas as well. This translation thus suffices to prove $L(\widetilde{\mathbf{\Omega}}_E) \leq_{\mathbb{N}} L(\widetilde{\mathbf{\Omega}}_{E'})$.

In the translation defined in the proof of Statement 2 if γ' has trivial domain and equality formulas then γ has trivial domain and equality formulas. Moreover, in this case for each $F \in \boldsymbol{F} \setminus \boldsymbol{F}'$ the counting operator appearing in η_F has trivial domain and equality formulas. This translation thus suffices to prove Statement 4.

□

Corollary 3.13. *Let $\mathbf{\Omega}$ be a set of* P*-bounded vectorised operators. There exists a set of almost relational* P*-bounded vectorised operators $\mathbf{\Omega}'$ such that for any* P*-bounded logic L that can simulate counting it follows that (i) $|\mathbf{\Omega}| = |\mathbf{\Omega}'|$, (ii) $L(\mathbf{\Omega}) \equiv_{\mathbb{N}} L(\mathbf{\Omega}')$, and (iii) $L(\widetilde{\mathbf{\Omega}}) \equiv_{\mathbb{N}} L(\widetilde{\mathbf{\Omega}}')$.*

Proof. Let L be a P-bounded logic that can simulate counting. For each $\mathbf{\Omega}_E \in \mathbf{\Omega}$ let $\mathbf{\Omega}_{E'}$ be the corresponding almost relational P-bounded vectorised operator in the statement of Proposition 3.12 and let $\mathbf{\Omega}'$ be the set of all such vectorised operators. It follows from the fact that $L(\mathbf{\Omega}_{E'}) \leq_{\mathbb{N}} L(\mathbf{\Omega}_E)$ that $\mathbf{\Omega}_{E'}$ is P-bounded. Statement (i) follows trivially. We can prove Statements (ii) and (iii) when $\mathbf{\Omega}$ is finite using a straightforward inductive argument.

To prove Statement (ii) and (iii) in the general case we note that each formula in a logic can reference only a finite number of vectorised operators.

□

We now prove the following useful normal form for formulas in a P-bounded extension of a logic by almost relational generalised operators.

Lemma 3.14. *Let L be a logic. Let $\mathbf{\Omega}$ be a set of almost relational generalised operators and suppose $L(\mathbf{\Omega})$ is* P*-bounded. For each $\theta(\vec{x}) \in L(\mathbf{\Omega})$ (resp. $\theta(\vec{x}) \in L(\widetilde{\mathbf{\Omega}})$) there is a formula $\theta'(\vec{x}) \in L(\mathbf{\Omega})$ (resp. $\theta'(\vec{x}) \in L(\widetilde{\mathbf{\Omega}})$) such that $\theta(\vec{x})$ and $\theta'(\vec{x})$ define the same query and such that for every sub-number-term γ of θ' if γ has a number-valued generalised operator at its head then γ appears only as an immediate sub-number-term of a subformula of θ' of the form $\vec{\mu} = \gamma$, where $\vec{\mu}$ is a sequence of number variables.*

Proof. Let L' be $L(\mathbf{\Omega})$ or $L(\widetilde{\mathbf{\Omega}})$. Let ϕ be an L'-formula. We call a number-term γ an *almost immediate sub-number-term* of ϕ if γ is a sub-number-term of ϕ but not a sub-number-term of any proper subformula of ϕ. We say that ϕ *satisfies the hypothesis* if for every sub-number-term γ of ϕ if γ has a number-valued generalised operator at its head then γ is an immediate sub-number-term of a subformula of ϕ of the form $\vec{\mu} = \gamma$, where $\vec{\mu}$ is a sequence of number variables.

Let $\theta(\vec{x})$ be an L'-formula and let $p(n) = n^t$ be a polynomial witnessing the fact that θ is P-bounded. We prove by structural induction that each subformula of θ' can be transformed into an equivilent formula that satisfies the hypothesis. Let $\phi(\vec{y})$ be a subformula of θ and suppose from the induction hypothesis we have that for every subformula ψ of ϕ there exists an L'-formula ψ' such that ψ and ψ' define the same query and ψ' satisfies the hypothesis. Let $\psi_1, \dots, \psi_k$ be the proper subformulas of ϕ such that each ψ_i is not a proper subformula of any proper subformula of ϕ. Let ϕ_0 be defined from ϕ by replacing each subformula ψ_i with the equivalent formula ψ_i' satisfying the hypothesis. Notice that ϕ_0 defines the same query as ϕ. If ϕ_0 satisfies the hypothesis let $\phi' := \phi_0$.

Suppose instead that ϕ_0 does not satisfy the hypothesis. It follows that there exists an immediate sub-number-term γ of ϕ_0 witnessing this fact. It follows that ϕ_0 does not have at its head an existential quantifier or universal quantifier, as otherwise it has no almost immediate sub-number-terms. Moreover, if ϕ_0 has a generalised operator at its head then γ cannot be a sub-number-term of any of the formulas that generalised operator is applied to. It follows from the fact that all generalised operators are almost relational that the variables that appear free in γ must appear free in ϕ_0.

We now define for each $j \in \mathbb{N}_0$ a formula ϕ_j by induction. We have already defined ϕ_0. For each $j \in \mathbb{N}_0$ let $\chi_1^j, \dots, \chi_{k_j}^j$ be the sub-number-terms of ϕ_j such that for each $i \in [k_j]$, χ_i^j has a number-valued generalised operator at its head and every other application of a number-valued generalised operator that appears in χ_i^j appears in a subformula of χ_i^j. For

each $i \in [k_j]$ let $\vec{\nu}_i^j$ be a fresh t-length sequence of number variables. Let ϕ_{j+1} be defined from ϕ_j by replacing each χ_i^j for $i \in [k_j]$ with the number term $\sum_{l \in [t]} \vec{\nu}_i^j(l) \cdot (\epsilon + 1)^{l-1}$.

Notice that if ϕ_j contains an almost immediate sub-number-term with a generalised operator at its head then ϕ_{j+1} contains strictly fewer almost immediate sub-number-terms with generalised operators at their heads. If ϕ_j does not contain an almost immediate sub-number-term with a generalised operator at its head then $\phi_j = \phi_{j+1}$. It follows that there exists some minimum $N \in \mathbb{N}$ such that for all $n > N$, $\phi_N = \phi_n$ and no almost immediate sub-number-term of ϕ_N has a generalised operator at its head. Let $\phi'_N := \phi_N$. We define ϕ'_j for each $j \in [N-1]$ by backwards induction as follows. Let

$$\phi'_j \equiv \exists \vec{\nu}_1^j, \ldots, \vec{\nu}_{k_j}^j \, [(\bigwedge_{i \in [k_j]} \vec{\nu}_i^j = \chi_i) \wedge \phi'_{j+1}]$$

and let

$$\phi' \equiv \exists \epsilon \, ((\forall \lambda \, (\lambda \leq \epsilon)) \wedge \phi'_1).$$

It is easy to see that ϕ' satisfies the hypothesis. It can be shown by induction that ϕ and ϕ' define the same query. It follows that there exists an L'-formula $\theta'(\vec{x})$ that defines the same query as $\theta(\vec{x})$ and satisfies the hypothesis.

□

We can associate with each number-valued generalised operator $\mathbf{\Omega}$ a Boolean-valued generalised operator $\mathbf{\Omega}^B$ defined by extending the vocabulary of $\mathbf{\Omega}$ with a constant symbol which is then used to denote a threshold. The evaluation function of $\mathbf{\Omega}^B$ is defined by comparing the output of the evaluation function of $\mathbf{\Omega}$ with the given threshold. We now define this notion formally.

Definition 3.15. Let Ω be a generalised operator. We now define a Boolean-valued generalised operator Ω^B. If Ω is a Boolean-valued generalised operator let $\Omega^B := \Omega$. Suppose instead that Ω is a number-valued generalised operator. Let $\tau = (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be the vocabulary of Ω. Let E be the evaluation function and ar be the arity of Ω. Let $\tau^B = (\boldsymbol{R}, \boldsymbol{F} \uplus \{t\}, \zeta^B)$ be such that t is a nullary constant symbol and $\zeta^B(T) = \zeta(T)$ for all $T \in \boldsymbol{R} \uplus \boldsymbol{F}$ and $\zeta^B(t) = ()$. Let $E^B : \mathrm{fin}^{\mathbb{N}}[\tau^B] \to \{0,1\}$ be defined for a τ^B-structure $\mathcal{A}$ such that $E^B(\mathcal{A}) = 1$ if, and only if, $E(\mathcal{A}^*) \geq t^{\mathcal{A}}$, where $\mathcal{A}^*$ is the reduct of $\mathcal{A}$ to τ. Let $\Omega^B = \Omega_{E^B,\mathrm{ar}}$. We call Ω^B the *set of Boolean-valued operators corresponding to* Ω. Let $\mathbf{\Omega}$ be a family of generalised operators. Let $\mathbf{\Omega}^B = \{\Omega^B : \Omega \in \mathbf{\Omega}\}$. We call $\mathbf{\Omega}^B$ the *set of Boolean-valued operators corresponding to* Ω.

Let $\mathbf{\Omega}$ be a P-bounded family of almost relational generalised operators. The normal form in Lemma 3.14 allows us to transform each formula in $L(\mathbf{\Omega})$ to one where each number-valued operator appears in a formula of the form '$\vec{\mu} = \mathbf{\Omega} \ldots$'. It is easy to see that we can replace each of these formulas with a conjunction of applications of $\mathbf{\Omega}^B$. We can conclude the following.

Lemma 3.16. *Let $\mathbf{\Omega}$ be a family of almost relational generalised operators. Let L be a logic and suppose $L(\mathbf{\Omega})$ is* P*-bounded. Then $L(\mathbf{\Omega}) \equiv L(\mathbf{\Omega}^B)$ and $L(\widetilde{\mathbf{\Omega}}) \equiv L(\widetilde{\mathbf{\Omega}}^B)$.*

3.3 Many-Sorted Quantifiers

In this section we show how to associate a generalised operator with a family of quantifiers so as to enable the translation from fixed-point logics to infinitary logics. We recall that generalised operators are defined in terms of *many-sorted* structures. We now generalise the notion of a Lindström quantifier and introduce the notion of a *many-sorted quantifier*.

Definition 3.17. Let $\tau := (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary. Let $\mathcal{G}$ be a class of τ-structures and let $\mathrm{ar} : \boldsymbol{S} \to \mathbb{N}_0$. We associate the pair $(\mathcal{G}, \mathrm{ar})$ with a *many-sorted quantifier* $Q_{\mathcal{G},\mathrm{ar}}$. We call $\mathcal{G}$ the *class of structures* and ar the *arity* of the quantifier $Q_{\mathcal{G},\mathrm{ar}}$. For a logic L the extension $L(Q_{\mathcal{G},\mathrm{ar}})$ is defined by extending the formula formation rules for L as follows:

> For each $s \in \boldsymbol{S}$ let $\vec{x}^s$, $\vec{x}^s_1$, and $\vec{x}^s_2$ be $\mathrm{ar}(s)$-tuples of element variables and let ϕ^D_s and $\phi^{\cong}_s$ be $L(Q_{\mathcal{G},\mathrm{ar}})$-formulas. For each $R \in \boldsymbol{R}$ and $i \in [r_R]$ let $\vec{x}^R_i$ be an $\mathrm{ar}(\zeta(R)(i))$-length tuple of element variables. For each $R \in \boldsymbol{R}$ let ϕ_R be an $L(Q_{\mathcal{G},\mathrm{ar}})$-formula. Then
>
> $$\phi \equiv Q_{\mathcal{G},\mathrm{ar}}[(\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\cong}_s)_{s\in\boldsymbol{S}}][(\vec{x}^R_1, \ldots, \vec{x}^R_{r_R})\phi_R]_{R\in\boldsymbol{R}}$$
>
> is a formula of $L(Q_{\mathcal{G},\mathrm{ar}})$. We have
>
> $$\mathrm{free}(\phi) = \bigcup_{s\in\boldsymbol{S}} [(\mathrm{free}(\phi^D_s) \setminus \vec{x}_s) \cup (\mathrm{free}(\phi^{\cong}_s) \setminus (\vec{x}^1_s \cup \vec{x}^2_s))] \cup \bigcup_{R\in\boldsymbol{R}} (\mathrm{free}(\phi_R) \setminus (\bigcup_{i\in[r_R]} \vec{x}^R_i)).$$

Let $\mathcal{I} := \langle (\phi^D_s)_{s\in\boldsymbol{S}}, (\phi^{\cong}_s)_{s\in\boldsymbol{S}}, (\phi_R)_{R\in\boldsymbol{R}} \rangle$. The semantics of the formula ϕ is defined for a structure $\mathcal{A} \in \mathrm{fin}[\rho]$ and assignment α to the free variables in ψ as follows

$$\mathcal{A} \models \psi[\alpha] \text{ if, and only if, } \mathcal{I}(\mathcal{A}, \alpha) \text{ is defined and } \mathcal{I}(\mathcal{A}, \alpha) \in \mathcal{G}$$

We note that many-sorted quantifiers can equivalently be thought of as Boolean-valued relational generalised operators that quantify over the universe. We also note that many-sorted quantifiers generalise Lindström quantifiers in the sense that we can identify each Lindström quantifier with a many-sorted quantifier with a single sort.

As for generalised operators, for a family of many-sorted quantifiers $\mathbf{Q}$ we let $L(\widetilde{\mathbf{Q}})$ be defined from $L(\mathbf{Q})$ by restricting ourselves to formulas in which each application of a many-sorted quantifier in $\mathbf{Q}$ has only trivial domain and equality formulas. In this case we omit the domain and equality formulas when applying the quantifier.

We aim to define a general method for associating a family of generalised operators with a family of many-sorted quantifiers. We do this in such a way as to enable a translation from each extension of fixed-point logic by a family of generalised operators to infinite families of first-order formulas extended by the corresponding family of many-sorted quantifiers. This translation generalises the translation from FPC to infinite families of FO+C-formulas (or just single $\mathcal{C}^\omega$-formulas) given by Grädel and Otto [21]. We now review this translation and then discuss what would be needed to generalise it.

Let $\theta(\vec{x})$ be an FPC-formula. We begin by unrolling the fixed-point operators in $\theta(\vec{x})$ in order to define a sequence of FOC-formulas $(\theta_n(\vec{x}))_{n\in\mathbb{N}}$, each of which capture the meaning of θ on structures of size n and such that there is a constant bound on the width of these formulas. For more details on this 'unrolling' argument please see [35]. We then aim to define an equivalent family of FO+C-formulas by defining a translation from FOC-formulas to FO+C-formulas. This translation involves recursively removing any reference to the number-sort and replacing instances of universal and existential quantification over the number domain with large disjunctions and conjunctions. To be precise, we recursively define for each $n \in \mathbb{N}$ and each subformula $\phi_n(\vec{y}, \vec{\nu})$ of θ_n a formula $\phi_{n,\vec{m}}(\vec{y})$ for each $\vec{m} \in \mathbb{N}^{|\vec{\nu}|}$ such that for any $\mathcal{A} \in \mathrm{fin}[\tau, n]$ and any assignment α to $\vec{y}$ in $\mathcal{A}$ we have $\mathcal{A} \models \phi_{n,\vec{m}}[\alpha]$ if, and only if, $\mathcal{A} \models \phi_n[\alpha\frac{\vec{m}}{\vec{\nu}}]$. In other words, we recursively define for each subformula a family of formulas, one for each assignment to the free number variables, each of which has the same meaning as the original subformula for the given assignment to the number variables.

We can translate a subformula with an existential or universal quantifier at its head that binds a number variable to a large disjunction or conjunction indexed by the possible assignments to that variable. For example, we translate a formula of the form $\psi_n(\vec{y}, \vec{\mu}) \equiv \exists\nu\, \phi_n(\vec{y}, \nu, \vec{\mu})$ to the family of formulas $(\psi_{n,\vec{m}})_{n\in\mathbb{N},\vec{m}\in[n]_0^{|\vec{\mu}|}}$ where each $\psi_{n,\vec{m}}(\vec{x}) \equiv \bigvee_{a\in[n]_0} \phi_{n,a,\vec{m}}(\vec{x})$. It is crucial at this point to note that we may assume without a loss of generality that every counting operator binds only a single *element* variable [21]. As such, we do not need to consider the problem of how to translate counting operators that bind number variables.

In contrast, an arbitrary generalised operator may bind element *and* number variables (e.g. the rank operator) and there is no known normal form that would allow us to restrict our attention to generalised operators that only bind element variables. As such, we cannot assume that all number variables are bound by universal or existential quantifiers and so we cannot handle the binding of number variables using disjunctions and conjunctions, and must consider how to translate applications of generalised operators that bind number variables. In order to address this we introduce for each operator a family of quantifiers each of which is applied to the entire family of formulas generated by the translation. Each of these quantifiers is defined over an extended vocabulary defined by 'copying' each relation symbol for each possible assignment to the bound number variables. We now define these quantifiers formally.

Let $\tau = (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary. Let $\mathcal{G}$ be a class of τ-structures, let $n \in \mathbb{N}$, and let $\mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}$. We associate $(\mathcal{G}, n, \mathrm{ar})$ with the many-sorted quantifier $Q_{\mathcal{G},n,\mathrm{ar}}$ defined as follows. Let

$$\boldsymbol{R}_{n,\mathrm{ar}} = \{R_{\vec{b}_1,\ldots,\vec{b}_{r_R}} : R \in \boldsymbol{R},\ \vec{b}_1 \in [n]_0^{\mathrm{ar}(\zeta(R)(1),2)}, \ldots, \vec{b}_{r_R} \in [n]_0^{\mathrm{ar}(\zeta(R)(r_R),2)}\},$$

where each $R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$ is a fresh relation symbol. Let $\zeta_{n,\mathrm{ar}}$ be defined such that $\zeta_{n,\mathrm{ar}}(R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}) = \zeta(R)$ for all $R_{\vec{b}_1,\ldots,\vec{b}_{r_R}} \in \boldsymbol{R}_{n,\mathrm{ar}}$. Let $\tau_{n,\mathrm{ar}} := (\boldsymbol{R}_{n,\mathrm{ar}}, \boldsymbol{S}, \zeta_{n,\mathrm{ar}})$. For each $\tau_{n,\mathrm{ar}}$-structure $\mathcal{A}$ of size n let $\uplus_{s\in\boldsymbol{S}} A_s$ be the universe of $\mathcal{A}$ and let $\mathcal{A}^*$ be the τ-structure defined as follows. For each $s \in \boldsymbol{S}$ let $A^*_s = A_s^{\mathrm{ar}(s,1)} \times [n]_0^{\mathrm{ar}(s,2)}$. For each $R \in \boldsymbol{R}$ let $R^{\mathcal{A}^*}$ be defined such that for all $i \in [r_R]$ and all $\vec{a}_i\vec{b}_i \in A^*_{\zeta(R)(i)}$, $(\vec{a}_1\vec{b}_1, \ldots, \vec{a}_{r_R}\vec{b}_{r_R}) \in R^{\mathcal{A}^*}$ if, and only if, $(\vec{a}_1, \ldots, \vec{a}_{r_R}) \in R^{\mathcal{A}}_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$. Let $\mathcal{A}^* = (\uplus_{s\in\boldsymbol{S}} A^*, (R^{\mathcal{A}^*})_{R\in\boldsymbol{R}})$. Let $\mathcal{G}_{n,\mathrm{ar}} := \{\mathcal{A} \in \mathrm{fin}[\tau_{n,\mathrm{ar}}] : \mathcal{A}^* \in \mathcal{G}\}$. Let $\mathrm{ar}^1 : \boldsymbol{S} \to \mathbb{N}_0$ be defined such that $\mathrm{ar}^1(s) = \mathrm{ar}(s, 1)$ for all $s \in \boldsymbol{S}$. Let $Q_{\mathcal{G},n,\mathrm{ar}} := Q_{\mathcal{G}_{n,\mathrm{ar}},\mathrm{ar}^1}$.

Notice that if $\mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0$ is such that $\mathrm{ar}(s, 2) = 0$ for all $s \in \boldsymbol{S}$ then for all $n \in \mathbb{N}$ we have $\tau_{n,\mathrm{ar}} = \tau$ and so $\mathcal{G}_{n,\mathrm{ar}} = \mathcal{G}$ and $Q_{\mathcal{G},n,\mathrm{ar}} = Q_{\mathcal{G},\mathrm{ar}}$.

Definition 3.18. Let $\tau = (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary. Let $\mathcal{G}$ be a class of τ-structures. We call the set of quantifiers $\mathbf{Q}_{\mathcal{G}} = \{Q_{\mathcal{G},\mathrm{ar},n} : n \in \mathbb{N},\ \mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0\}$ the *vectorised family of many-sorted quantifiers* (or just the *vectorised many-sorted quantifier*) *generated* by $\mathcal{G}$.

We now define for each almost relational Boolean-valued generalised operator Ω a corepsonding family of many-sorted quantifiers $\mathbf{Q}_{\boldsymbol{\Omega}}$. Let $\Omega := \Omega_{E,\mathrm{ar}}$ be an almost relational Boolean-valued generalised operator and let $\tau := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be the vocabulary of Ω. We note that since Ω is almost relational it follows that each function symbol $F \in\in \boldsymbol{F}$ is a constant symbol. Let $\tau_{\mathbf{rel}} := (\boldsymbol{R}, \boldsymbol{S}, \zeta|_{\boldsymbol{R}})$. For each $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ and $\mathcal{A} \in \mathrm{fin}[\tau_{\mathbf{rel}}]$ let $(\mathcal{A}|\alpha)$ be the τ-structure $(U(\mathcal{A}), (R^{\mathcal{A}})_{R\in\boldsymbol{R}}, (\alpha(F))_{F\in\boldsymbol{F}})$. We can think of each $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ as an assignment to the constant symbols and each τ-structure $(\mathcal{A}|\alpha)$ as an expansion of $\mathcal{A}$ with constant symbols $\boldsymbol{F}$ assigned according to α. For each $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ let $\mathcal{G}_{E,\alpha} := \{\mathcal{A} \in \mathrm{fin}[\tau_{\mathbf{rel}}] : E(\mathcal{A}|\alpha) = 1\}$. In order to avoid excessive subscripts we write $Q_{E,\alpha,n,\mathrm{ar}}$ to denote the quantifier $Q_{\mathcal{G}_{E,\alpha},n,\mathrm{ar}}$ for each $n \in \mathbb{N}$. Let

$$\mathbf{Q}_{\Omega} := \{Q_{E,\alpha,n,\mathrm{ar}} : n \in \mathbb{N}, \alpha : \boldsymbol{F} \to \mathbb{N}_0\}.$$

and let

$$\mathbf{Q}_{E,\alpha} := \mathbf{Q}_{\mathcal{G}_{E,\alpha}} = \{Q_{E,\alpha,n,\mathrm{ar}} : n \in \mathbb{N}, \mathrm{ar} : \boldsymbol{S} \times [2] \to \mathbb{N}_0\}.$$

We call $\mathbf{Q}_{\Omega}$ *the set of quantifiers correpsonding to* Ω. We note that when Ω is relational $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ is the empty function and $\mathcal{A} = (\mathcal{A}|\alpha)$. In this case we omit α in the subscript. Let Ω be an almost relational generalised operator. We let $\mathbf{Q}_{\Omega} := \mathbf{Q}_{\Omega^B}$ and we call $\mathbf{Q}_{\Omega}$

the set of quantifiers correpsonding to Ω. Let $\mathbf{\Omega}$ be a set of almost relational generalised operators. Let

$$\mathbf{Q}_{\mathbf{\Omega}} = \bigcup_{\Omega \in \mathbf{\Omega}} \mathbf{Q}_{\Omega}.$$

We call $\mathbf{Q}_{\mathbf{\Omega}}$ *the set of quantifiers corresponding to* $\mathbf{\Omega}$. In order to illustrate the idea behind this definition we consider the counting operator as an example and construct the corresponding set of quantifiers. We see that the set of counting quantifiers corresponds to the counting operator.

Example 3.19. We recall that the counting operator $\mathbf{\Omega}_{E_{\mathbf{cnt}}}$ has the vocabulary $\tau_{\mathbf{set}} = (\{U\}, \emptyset, \{s\}, \zeta)$. The corresponding set of Boolean-valued generalised operators $\mathbf{\Omega}_{E^B_{\mathbf{cnt}}}$ has the vocabulary $\tau^B_{\mathbf{set}} = (\{U\}, \{t\}, \{s\}, \zeta^B)$, where t is some constant function symbol. From [21] it follows that $\mathrm{FPC} \equiv \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}_{E_{\mathbf{cnt}}}) \equiv \mathrm{FP}^{\mathbb{N}}(\Omega_{E_{\mathbf{cnt}},\mathrm{ar}})$ where $\mathrm{ar} : \{s\} \times [2] \to \mathbb{N}_0$ is defined such that $\mathrm{ar}(s,1) = 1$ and $\mathrm{ar}(s,2) = 0$. We construct the set of quantifiers corresponding to $\Omega_{E_{\mathbf{cnt}},\mathrm{ar}}$. Let $\alpha : \{t\} \to \mathbb{N}_0$ and let $\mathcal{G} := \mathcal{G}_{E^B_{\mathbf{cnt}},\alpha}$. Since $\mathrm{ar}(s,2) = 0$ it follows that for all $n \in \mathbb{N}$ we have $\mathcal{G}_{n,\mathrm{ar}} = \mathcal{G}$ and $Q_{E^B_{\mathbf{cnt}},\alpha,n,\mathrm{ar}} = Q_{\mathcal{G},n,\mathrm{ar}} = Q_{\mathcal{G},\mathrm{ar}^1}$, where $\mathrm{ar}^1 : \{s\} \to \mathbb{N}_0$ is defined such that $\mathrm{ar}^1(s) = 1$. Recall that $\mathcal{G}$ is defined such that for any $\mathcal{A} \in \mathrm{fin}[\tau_{\mathbf{set}}]$, $\mathcal{A} \in \mathcal{G}$ if, and only if, $|U^{\mathcal{A}}| \geq \alpha(t)$. The Lindström quantifier $\exists^{\geq \alpha(t)}$ is also defined by $\mathcal{G}$ and so we can identify $Q_{\mathcal{G},\mathrm{ar}^1}$ with $\exists^{\geq \alpha(t)}$. It follows that for all $n \in \mathbb{N}$ and $\alpha : \{t\} \to \mathbb{N}_0$ we can identify $Q_{E^B_{\mathbf{cnt}},\alpha,n,\mathrm{ar}}$ with the counting quantifier $\exists^{\geq \alpha(t)}$ and so

$$\begin{aligned}\mathbf{Q}_{\Omega_{E_{\mathbf{cnt}},\mathrm{ar}}} = \mathbf{Q}_{\Omega_{E^B_{\mathbf{cnt}},\mathrm{ar}}} &= \{Q_{E^B_{\mathbf{cnt}},\alpha,n,\mathrm{ar}} : n \in \mathbb{N}, \alpha : \{t\} \to \mathbb{N}_0\} \\ &= \{\exists^{\geq k} : k \in \mathbb{N}_0\}.\end{aligned}$$

3.4 Translating Formulas to Substitution Programs

There is a standard translation from FP to $\mathcal{L}^\omega$ given by [35]. This translation involves defining for each FP-formula θ a family of FO-formulas $(\theta_n)_{n\in\mathbb{N}}$ where each θ_n has the same meaning as θ for structures of size n. For every $n \in \mathbb{N}$ we define θ_n by replacing each application of the fixed-point operator in θ with an explicit implementation in first-order logic of the construction of an inflationary fixed-point. This involves for each fixed-point operator recursively replacing each occurrence of the bound second-order variable with its inductive definition. However, if within the scope of a fixed-point operator the bound second-order variable appears more than once then each recursive step results in a doubling of the size of the formula. It follows that, in general, the family of formulas $(\theta_n)_{n\in\mathbb{N}}$ has exponential-size.

In contrast, the standard translation from FP to families of symmetric circuits given by [32] associates each FP-formula with a *polynomial-size* family of symmetric circuits defined over the standard basis. These two translations use similar approaches and both involve

unrolling the fixed-point operator in the sense discussed above. However, the important difference is that when we define a circuit we can reuse the output of any gate. In particular, we can reuse the recursive definition of the second-order variable multiple times without imposing an additional cost.

In this section we introduce the notion of a substitution program. We think of a substitution program as a more succinct representation of a formula, one in which we are allowed to 'name' subformulas and then refer to them without having to rewrite their definition. In this way a substitution program is very similar to a circuit. Indeed, we use substitution programs both as an intermediary step when we define translations from fixed-point logics to infinitary logics in Section 3.5 and when we define translations from fixed-point logics to families of symmetric circuits in Chapter 8.

Definition 3.20. Let L be a logic and ρ be a relational vocabulary. An $L[\rho]$-*substitution program* is a sequence of $L[\rho]$-formulas $\Phi := (\phi_1(\vec{x}_1, \vec{\mu}_1); \boldsymbol{V}_1), \ldots, \phi_k(\vec{x}_k, \vec{\mu}_k); \boldsymbol{V}_k))$ such that for each $i \in [k]$

- $\vec{x}_i$ is a sequence of element-variables and $\vec{\mu}_i$ is a sequence of number-variables,
- if $i > 1$ then V_i is a mixed-sort second-order variable that has the same type as $(\vec{x}_i, \vec{\mu}_i)$, and
- $\boldsymbol{V}_i \subseteq \{V_j : i < j \leq k\}$.

For each $i \in [k]$ the *flattening* of Φ at i is defined recursively as follows. If $i = k$ then the flattening of Φ at i is ϕ_i. Suppose $i < k$ and for each $j \in [i+1, k]$ let $\psi_j(\vec{x}_j, \vec{\mu}_j)$ be the flattening of Φ at j. The flattening of Φ at i is defined by replacing each subformula in ϕ_i of the form $V_j(\vec{y}, \vec{\nu})$, where $V_j \in \boldsymbol{V}_i$, with the formula

$$\exists \vec{z} \vec{\eta} [(\vec{z} = \vec{y} \wedge \vec{\eta} = \vec{\nu}) \wedge \exists \vec{x}_j \vec{\mu}_j (\vec{x}_j = \vec{z} \wedge \vec{\mu}_j = \vec{\eta} \wedge \psi_j(\vec{x}_j \vec{\mu}_j))].$$

The *flattening* of Φ is the flattening of Φ at 1. The free variables of Φ are the free variables in the flattening of Φ. For sequences of variables $\vec{x}, \vec{\mu}$ we write $\Phi(\vec{x}, \vec{\mu})$ to indicate that the free variables in Φ are among $\vec{x}, \vec{\mu}$. Let $\mathcal{A} \in \text{fin}[\rho]$ and let α be an assignment to the free variables in Φ in $\mathcal{A}$. Let ϕ be the flattening of Φ. We say $\mathcal{A} \models \Phi[\alpha]$ if, and only if, $\mathcal{A} \models \phi[\alpha]$. If all of the free variables of Φ are element-variables we say that Φ defines a query Q if the flattening of Φ defines the query Q. The *formula length* of a substitution program Φ is the maximal length of a formula in Φ. The *width* of a substitution program is the maximum width of a formula in Φ.

Example 3.21. Let $\tau := \{E, A\}$ be a vocabulary. We think of a τ-structure G as a directed graph with all vertices in the unary relation A labelled by a $\forall$ symbol. We define the alternating path relation recursively as follows. We say there is an alternating path from a to b in G if (i) $a = b$, (ii) a is not in A^G and there exists a vertex c adjacent to a such that

there is an alternating path from c to b, or (iii) a is in A^G and for every vertex c adjacent to a there is an alternating path from c to b.

We now define a family of substitution programs intended to define the alternating path relation. Let $n \in \mathbb{N}$ and let $\Phi_n := (\phi_i(x_i, y_i))_{i \in [n^2]}$ where $\phi_{n^2}(x_{n^2}, y_{n^2}) :\equiv x_{n^2} = y_{n^2}$ and for all $i < n^2$

$$\phi_i(x_i, y_i) :\equiv x_i = y_i \vee [\exists z(E(x_i, z) \wedge V_{i+1}(z, y_i)) \\ \wedge A(x_i) \implies \forall z(E(x_i, z) \implies V_{i+1}(z, y_i))].$$

We think of each formula in the substitution program Φ_n as denoting a single recursive step in the definition of the alternating path relation for τ-structures of size n. It can be shown that the function $n \mapsto \Phi_n$ can be computed in time polynomial in n. The flattening of Φ_n is given by starting with ϕ_1 and then recursively replacing each relation V_i with its definition ϕ_i (while renaming variables). The flattening of Φ_n is thus a single first-order formula that defines the alternating path relation for structures of size n. Notice that in each step in the definition of the flattening of Φ_n we replace each of the two occurrences of the second-order variable with its definition. It is not hard to see that the resultant flattenings, in contrast with the substitution programs, will have size exponential in n.

Remark 3.22. A substitution program is a finite linearly ordered set of formulas, and we index this sequence by an initial segment of the natural numbers. It is known that any finite linearly ordered set can be identified with an initial segment of the natural numbers and with this in mind we will sometimes index a substitution program by some linearly ordered set other than the natural numbers rather than explicitly include an order-preserving bijection.

There is a notion of the width of a formula that measures the maximal number of variables that appear free in any subformula. In some cases we are not just interested in the number of free variables that appear free in a given subformula but rather the number of variables that are introduced into the scope by an operator or quantifier. Let Ω be a generalised operator with arity ar and vocabulary $\tau := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$. The *operator width* of Ω is equal to $\max\{\sum_{i \in [r_R]}(\mathrm{ar}(\zeta(R)(i), 1) + \mathrm{ar}(\zeta(R)(i), 2)) : R \in \boldsymbol{R}\}$. We can similarly regard many-sorted quantifiers as generalised operators and so associate each of them with an operator width. We say that the universal and existential quantifiers each have operator width 1. Let L be a logic and let $\boldsymbol{\Omega}$ be a set of generalised operators. Let θ be an $L(\boldsymbol{\Omega})$-formula. The *operator width* of θ is equal to the maximal operator width of any quantifier or generalised operator that appears in θ. We denote the operator width of θ by $\mathrm{owidth}(\theta)$. The operator width of a $L(\boldsymbol{\Omega})[\rho]$-substitution program Φ is equal to the maximal operator width of any of formula in Φ. It is not hard to see that any formula can be transformed into a normal form with the property that the operator width of that formula is at most equal to its width.

Definition 3.23. Let ρ be a relational vocabulary and let L be a logic. Let $\vec{x}$ be a sequence of element variables and let $\vec{\mu}$ be sequence of number variables. Let $\mathbf{\Phi} := (\Phi_n(\vec{x}, \vec{\mu}))_{n\in\mathbb{N}}$ be a family of $L[\rho]$-substitution programs. We say $\mathbf{\Phi}$ is a P-*uniform family of substitution programs* if the function $n \mapsto \Phi_n(\vec{x}, \vec{\mu})$ is computable in time polynomial in n. We say that $\mathbf{\Phi}$ has *constant width* if there exists $k \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ the width and operator width of Φ_n is at most k. We say that $\mathbf{\Phi}$ has *constant length* if there exists $k \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ the formula length of Φ_n is at most k.

Suppose for all $n \in \mathbb{N}$ that all of the free variables in Φ_n are element variables (i.e. $\vec{\mu} = ()$) and let Q_n be the query defined by Φ_n. Let Q be the query such that $Q(\mathcal{A}) = Q_{|\mathcal{A}|}(\mathcal{A})$ for each $\mathcal{A} \in \mathrm{fin}[\rho]$. We call Q *the query defined by* $\mathbf{\Phi}$.

For any given formula θ in an extension of a fixed-point logic we can unroll the fixed points and so define a family $(\theta_n)_{n\in\mathbb{N}}$ of formulas without any application of the fixed-point operator and such that for each $n \in \mathbb{N}$, θ_n and θ define the same query on structures of size n. This approach can also be used to prove the following result. We omit the proof here as the technique is standard. For more detail please see [35].

Lemma 3.24. *Let $\mathbf{\Omega}$ be a set of generalised operators. If a query can be defined in* $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ *(resp.* $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$*), then it can be defined by a* P*-uniform family of* $\mathrm{FO}^{\mathbb{N}}(\mathbf{\Omega})$*-substitution programs (resp.* $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$*-substitution programs) with constant length.*

We aim to show that every P-uniform family of $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$-substitution programs with constant length can be translated to an equivalent P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}})$-substitution programs with constant width. We prove this result in two stages. Let $\mathbf{\Theta} = (\Theta_n)_{n\in\mathbb{N}}$ be a P-uniform family of $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$-substitution programs with constant length. First, we show in Lemma 3.25 that for every $n \in \mathbb{N}$ each formula $\theta_{n,i}(\vec{x}_i, \vec{\mu}_i)$ in Θ_n can be associated with a family of formulas $(\theta_{n,i;\beta}(\vec{x}_i))_{\beta\in[n]_0^{\vec{\mu}_i}}$ in $\mathrm{FO}(\widetilde{\mathbf{Q}})$ each of which capture the meaning of $\theta_{n,i}$ on structures of size n when the number variables are assigned according to β. Second, we show in Lemma 3.26 that by concatenating these families we can form a single P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}})$-substitution programs $\mathbf{\Theta}'$ with constant width such that $\mathbf{\Theta}$ and $\mathbf{\Theta}'$ define the same query.

We now formalise this argument. Let ρ be a relational vocabulary and let $\mathbf{\Omega}$ be a set of almost relational generalised operators. Let $\mathbf{Q} = \mathbf{Q}_{\mathbf{\Omega}}$. Let $\theta(\vec{x}, \vec{\mu}; \vec{V})$ be a formula in $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$ where $\vec{V} = (V_1, \ldots, V_v)$ is a sequence of mixed-sort second-order variables. For each $i \in [v]$ let v_i and m_i be the arity of the element-sort and number-sort of V_i, respectively. Let $n \in \mathbb{N}$. For each $i \in [v]$ and $\vec{c} \in [n]_0^{m_i}$ let $V_{i,\vec{c}}^n$ be an element-sort second-order variable with arity v_i. Let $\vec{W}^n := \{V_{i,\vec{c}}^n : i \in [v], \vec{c} \in [n]_0^{m_i}\}$ and let $\boldsymbol{\theta}_n = (\theta_{n;\beta}(\vec{x}, \vec{W}^n))_{\beta\in[n]_0^{\vec{\mu}}}$ be a sequence of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formulas. We say $\boldsymbol{\theta}_n$ *translates* θ *for* n if for all

1. $\mathcal{A} \in \mathrm{fin}[\rho, n]$;
2. assignments $\alpha \in A^{\vec{x}}$ and $\beta \in [n]_0^{\vec{\mu}}$; and

3. assignments γ and γ' to the second-order variables, where γ maps each V_i to a relation $V_i^{(\mathcal{A},\gamma)} \subseteq A^{v_i} \times [n]_0^{m_i}$ and γ' maps each $V_{i,\vec{c}}^n$ to the relation $(V_{i,\vec{c}}^n)^{(\mathcal{A},\gamma')} \subseteq A^{v_i}$ such that $(\vec{a},\vec{c}) \in V_i^{(\mathcal{A},\gamma)}$ if, and only if, $\vec{a} \in (V_{i,\vec{c}}^n)^{(\mathcal{A},\gamma')}$,

we have

$$\mathcal{A} \models \theta[\alpha,\beta,\gamma] \iff \mathcal{A} \models \theta_{n;\beta}[\alpha,\gamma'].$$

We say $\boldsymbol{\theta} := (\boldsymbol{\theta}_n)_{n\in\mathbb{N}} = (\theta_{n;\beta})_{n\in\mathbb{N},\beta\in[n]_0^{\vec{\mu}}}$ *translates* θ if $\boldsymbol{\theta}_n$ translates θ for all $n \in \mathbb{N}$.

Lemma 3.25. *Let ρ be a relational vocabulary. Let $\boldsymbol{\Omega}$ be a set of almost relational Boolean-valued generalised operators and suppose $\mathrm{FO}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}})$ is* P*-bounded. Let $\theta(\vec{x},\vec{\mu};\vec{V}) \in \mathrm{FO}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}})[\rho]$ where $\vec{V} = (V_1,\dots,V_v)$ is a sequence of mixed-sort second-order variables.*

There exists a sequence of $\mathrm{FO}(\widetilde{\mathbf{Q}}_{\boldsymbol{\Omega}})[\rho]$-formulas $\boldsymbol{\theta} := (\boldsymbol{\theta}_n)_{n\in\mathbb{N}} = (\theta_{n;\beta}(\vec{x};\vec{W}^n))_{n\in\mathbb{N},\beta\in[n]_0^{\vec{\mu}}}$ such that $\boldsymbol{\theta}$ translates θ. Each formula in $\boldsymbol{\theta}$ has width at most equal to the element-variable width of θ. Moreover, there is a constant c such that the function that maps (n,θ) to $(\theta_{n;\beta})_{\beta\in[n]_0^{\vec{\mu}}}$ is computable in time $\mathcal{O}((|\theta|+n)^{c|\mathrm{cl}(\theta)|})$.

Proof. Let $n \in \mathbb{N}$. We aim to prove by induction that for every subformula $\phi(\vec{y},\vec{\nu})$ of θ there exists a family $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formulas $\boldsymbol{\phi}_n = (\phi_{n;\beta}(\vec{y};\vec{W}^n))_{\beta\in[n]_0^{\vec{\nu}}}$ that translate ϕ for n, where $\vec{W}^n = \{V_{i,\vec{a}}^n : i \in [v], \vec{a} \in [n]_0^{m_i}\}$.

Let $\phi(\vec{y},\vec{\nu})$ be a subformula of θ and let $\beta \in [n]_0^{\vec{\nu}}$. It follows from the induction hypothesis that for each subformula $\psi(\vec{z},\vec{\lambda})$ of ϕ there exists a family of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formulas $\boldsymbol{\psi}_n = (\psi_{n;\beta}(\vec{z};\vec{W}^n))_{\beta\in[n]_0^{\vec{\lambda}}}$ that translate ψ for n. We now consider those cases for which ϕ is an atomic formula. If ϕ is an atomic formula and ϕ does not have a generalised operator at its head then let $\phi_{n;\beta} \equiv \phi$. If $\phi \equiv V_i(\vec{y},\vec{\nu})$ for some $i \in [v]$ let $\phi_{n;\beta}(\vec{y}) \equiv V_{i,\beta(\vec{\nu})}^n(\vec{y})$. Suppose $\phi \equiv \gamma_1 \leq \gamma_2$ for two number-terms γ_1 and γ_2. Since all of the generalised operators in $\boldsymbol{\Omega}$ are Boolean-valued it follows that every sub-number-term of θ has no subformulas or free element variables. It follows that the evaluation of a sub-number-term in θ depends just on the assignment to its free (number) variables and the size of the structure over which it is being evaluated. We can compute the values of γ_1 and γ_2 for the assignment β and a structure of size n. Let t_1 be the value of γ_1 and t_2 be the value of γ_2. Let $\phi_{n;\beta} \equiv \forall y\,(y = y)$ if $t_1 \leq t_2$ and otherwise let $\phi_{n;\beta} :\equiv \exists x\,(x \neq x)$. We handle the case where $\phi \equiv \gamma_1 = \gamma_2$ similarly. Suppose ϕ is of the form

$$\phi \equiv \Omega_{E,\mathrm{ar}}[(\vec{x}_1^R\vec{\mu}_1^R,\dots,\vec{x}_{r_R}^R\vec{\mu}_{r_R}^R)\,\psi_R]_{R\in\boldsymbol{R}}[\eta_F]_{F\in\boldsymbol{F}}.$$

Let $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ be defined such that $\alpha(F)$ is the value of η_F for structures of size n with assignment β. Let $\tau := (\boldsymbol{R},\boldsymbol{F},\boldsymbol{S},\zeta)$ be the vocabulary of $\Omega_{E,\mathrm{ar}}$. From the induction hypothesis there exists for each $R \in \boldsymbol{R}$ a family of formulas

$$\boldsymbol{\psi}_{R,n} = \{\psi_{R,n;\beta}((\vec{x}_1^R,\dots,\vec{x}_{r_R}^R,\vec{y};\vec{W}^n) : \beta \in [n]_0^{\vec{\mu}_1^R\cup\dots\cup\vec{\mu}_{r_R}^R\cup\vec{\nu}}\}$$

such that $\boldsymbol{\psi}_{R,n}$ translates ψ_R for n. We recall that the quantifier $Q_{E,\alpha,n,\mathrm{ar}}$ has the vocabulary $\tau_{n,ar} = (\boldsymbol{R}_{n,\mathrm{ar}}, \boldsymbol{S}, \zeta_{n,\mathrm{ar}})$. For each $R_{\vec{b}_1,\dots,\vec{b}_{r_R}} \in \boldsymbol{R}_{n,\mathrm{ar}}$ let $\psi_{R_{\vec{b}_1,\dots,\vec{b}_{r_R}}} := \psi_{R,n;\beta'}$ where $\beta' = \beta\frac{\vec{b}_1}{\vec{\mu}_1^R}\dots\frac{\vec{b}_{r_R}}{\vec{\mu}_{r_R}^R}$. Let

$$\phi_{n;\beta} \equiv Q_{E,\alpha,n,\mathrm{ar}}[(\vec{x}_1^R,\dots,\vec{x}_{r_R}^R)\,\psi_{R_{\vec{b}_1,\dots,\vec{b}_{r_R}}}]_{R_{\vec{b}_1,\dots,\vec{b}_{r_R}}\in\boldsymbol{R}_{n,\tau}}.$$

Suppose $\phi(\vec{y},\vec{\nu}) \equiv \psi_1(\vec{y}_1,\vec{\nu}_1) \wedge \psi_2(\vec{y}_2,\vec{\nu}_2)$. From the induction hypothesis there exists $\boldsymbol{\psi}_{1,n} = (\phi_{1,n;\beta}(\vec{y};\vec{W}^n))_{\beta\in[n]_0^{\vec{\nu}_1}}$ and $\boldsymbol{\psi}_{2,n} = (\phi_{2,n;\beta}(\vec{y};\vec{W}^n))_{\beta\in[n]_0^{\vec{\nu}_2}}$ such that $\boldsymbol{\psi}_{1,n}$ translates ψ_1 for n and $\boldsymbol{\psi}_{2,n}$ translates ψ_2 for n. Let β_1 be the restriction of β to $\vec{\nu}_1$ and let β_2 be the restriction of β to $\vec{\nu}_2$. Let

$$\phi_{n;\beta} \equiv \psi_{1,n;\beta_1} \wedge \psi_{2,n;\beta_2}.$$

The other logical connectives can be handled similarly. Suppose $\phi(\vec{y},\vec{\nu}) \equiv \exists x\,\psi$, where x is a number variable. Let $\psi_{n;\beta} \equiv \exists x\,\psi_{n;\beta}$. Suppose $\phi(\vec{y},\vec{\nu}) \equiv \exists\lambda\,\psi$, where λ is a number variable. Let

$$\psi_{n;\beta} \equiv \bigvee_{a\in[n]_0} \psi_{n;\beta_a}(\vec{y}),$$

where for each $a \in [n]_0$, $\beta_a = \beta\frac{a}{\lambda}$. We can handle the universal quantifier similarly using conjunctions instead of disjunctions.

It is not hard to show that in each of these cases the constructed families of formulas satisfy the requirements for a translation. This completes the inductive argument.

It can be shown that there is an algorithm that takes as input a number-term γ with no subformulas, a number $n \in \mathbb{N}$, and an assignment β that maps the free (number) variables of γ to $[n]_0$ and outputs the value of γ for any structure of size n with the assignment β and runs in time polynomial in n and the length of γ. Let $p(n) = an^{c_1}$ be a polynomial bounding the running time of this algorithm, where a and c_1 are constants.

Let $n \in \mathbb{N}$ and $\theta(\vec{x},\vec{\mu};\vec{V}) \in \mathrm{FO}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}})[\rho]$. We can construct a translation of θ for n by implementing the above inductive argument as a recursive algorithm. Moreover, it can be shown that there exists constants c_2 and c_3 such that we can construct a translation of a subformula ϕ for n in time at most $c_2\cdot(|\phi|+n)^{c_1}\cdot n^{c_3\cdot(\mathrm{width}(\phi)+|\phi|)}\cdot X_\phi$ where X_ϕ is the maximal cost of translating an immediate subformula ϕ. It follows that we can compute a translation for θ for n in time at most $c_2^{|\mathrm{cl}(\theta)|}(|\theta|+n)^{c_1\cdot|\mathrm{cl}(\theta)|}n^{c_3\cdot|\mathrm{cl}(\theta)|\cdot(w_\theta+|\theta|)}$ and so, since $|\mathrm{cl}(\theta)| \le |\theta|$, in time $\mathcal{O}((n+|\theta|)^{c\cdot|\theta|})$ for some constant c.

□

Lemma 3.26. *Let ρ be a relational vocabulary and let $\boldsymbol{\Omega}$ be a set of almost relational Boolean-valued operators such that $\mathrm{FO}^{\mathbb{N}}(\widetilde{\boldsymbol{\Omega}})$ is* P*-bounded. Let $\mathbf{Q} = \mathbf{Q}_{\boldsymbol{\Omega}}$. Every query definable by a*

P-*uniform family of* $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$-*substitution programs with constant length is definable by a* P-*uniform family of* $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-*substitution programs with constant width.*

Proof. Let $\Theta := (\Theta_n)_{n\in\mathbb{N}}$ be a P-uniform family of $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$-substitution programs with constant length. For each $n \in \mathbb{N}$ let M_n be the domain of the function Θ_n. For each $n \in \mathbb{N}$, $i \in M_n$ let $\vec{x}_i^n$ and $\vec{\mu}_i^n$ be the free element variables and number variables in $\Theta_n(i)$ and let V_i^n be the second-order variable associated with $\Theta_n(i)$. Note that V_i^n has the same type as $(\vec{x}_i^n, \vec{\mu}_i^n)$. For each $n \in \mathbb{N}$, $i \in M_n$, and $\vec{c} \in [n]_0^{|\vec{\mu}_i^n|}$ let $V_{i,\vec{c}}^n$ be an element-sort second-order variable with arity $|\vec{x}_i^n|$ and let $\theta_{n,i}(\vec{x}_i^n, \vec{\mu}_i^n; \vec{V}_i^n) := \Theta_n(i)$, where $\vec{V}_i^n = (V_j^n)_{j>i}$. It follows from Lemma 3.25 that there exists a family of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formulas $\boldsymbol{\theta}_{n,i} := (\theta_{n,i;\beta}(\vec{x}_i^n; \vec{W}_i^n))_{\beta\in[n]_0^{\vec{\mu}_i^n}}$ that translates $\theta_{n,i}$ for n and where $\vec{W}_i^n = \{V_{j,\vec{c}}^n : j \in M_n, j > i, \vec{c} \in [n]_0^{|\vec{\mu}_j^n|}\}$.

Let $M_n' := \{(i, \vec{c}) : i \in M_n, \vec{c} \in [n]_0^{|\vec{\mu}_i^n|}\}$. Let $\Theta_n' : M_n' \to \mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$ be defined such that for each $(i, \vec{c}) \in M_n'$ let $\beta = \frac{\vec{c}}{\vec{\mu}_i^n}$ then we have $\Theta_n'(i, \vec{c}) := \theta_{n,i;\beta}$. Let $\Theta' = (\Theta_n')_{n\in\mathbb{N}}$. Since M_n' is a product of two linearly ordered sets, we can define a linear order on M_n' lexicographically. For each $(i, \vec{c}) \in M_i'$ we associate the formula $\Theta_n'(i, \vec{c})$ with the second-order variable $V_{i,\vec{c}}^n$. For each $(i, \vec{c}) \in M_i'$ the second-order variables that appear in $\Theta_n'(i, \vec{a})$ are among $\vec{W}_i^n \subseteq \{V_{j,\vec{b}}^n : (j, \vec{b}) \in M_i', (j, \vec{b}) > (i, \vec{c})\}$. It follows that Θ_n' is a valid substitution program.

Claim 3.26.1. *For each $n \in \mathbb{N}$ and $i \in M_n$ let $\phi_{n,i}(\vec{x}_i^n, \vec{\mu}_i^n)$ be the flattening of Θ_n at i and for each $\vec{c} \in [n]_0^{|\vec{\mu}_i^n|}$ let $\phi'_{n,(i,\vec{c})}(\vec{x}_i^n)$ be the flattening of Θ_n' at $(i, \vec{c})$. Then for all $n \in \mathbb{N}$ and $i \in M_n$ $\mathcal{A} \in \mathrm{fin}[\rho, n]$, $\alpha \in A^{\vec{x}_i^n}$, and $\beta \in [n]_0^{\vec{\mu}_i^n}$ we have $\mathcal{A} \models \phi_{n,i}[\alpha, \beta]$ if, and only if, $\mathcal{A} \models \phi'_{n,(i,\beta(\vec{\mu}_i^n))}[\alpha]$.*

Proof. Let $n \in \mathbb{N}$. We prove this result by backwards induction. Let $i \in M_n$. Suppose $i = \max(M_n)$. Then no second-order variables appear in $\Theta_n(i)$ and for all $\vec{c} \in [n]_0^{|\vec{\mu}_i^n|}$ no second-order variables appear in $\Theta_n'(i, \vec{c})$. It follows that for all $\vec{c} \in [n]_0^{|\vec{\mu}_i^n|}$ we have $\phi_{n,i} = \Theta_n(i)$ and $\phi'_{n,(i,\vec{c})} = \Theta_n'(i, \vec{c})$. We thus have $\mathcal{A} \models \phi_{n,i}[\alpha, \beta]$ if, and only if, $\mathcal{A} \models \Theta_n(i)[\alpha, \beta]$ if, and only if, $\mathcal{A} \models \theta_{n,i;\beta}[\alpha]$ if, and only if, $\mathcal{A} \models \Theta_n'(i, \beta(\vec{\mu}_i^n))[\alpha]$ if, and only if, $\mathcal{A} \models \phi'_{n,(i,\beta(\vec{\mu}_i^n))}[\alpha]$. The second equivalence follows from the definition of a translation and from the fact that no second-order variables appear in either of these formulas. The base case follows.

Suppose $i < \max(M_n)$ and the hypothesis holds for all $j > i$. Let γ be an assignment to second-order variables that maps each V_j^n with $j > i$ to the relation $(V_j^n)^{(\mathcal{A},\gamma)} := \phi_{n,j}^{\mathcal{A}}$. Let γ' be the assignment to second-order variables that maps each $V_{j,\vec{c}}^n$ with $j > i$ and $\vec{c} \in [n]_0^{|\vec{\mu}_j^n|}$ to the relation $(V_{j,\vec{c}}^n)^{(\mathcal{A},\gamma')} := (\phi'_{n,(j,\vec{c})})^{\mathcal{A}}$. From the induction hypothesis we have for all $j > i$, $\alpha_j \in A^{\vec{\mu}_j^n}$, and $\beta_j \in [n]_0^{\vec{\mu}_j^n}$ that $\mathcal{A} \models \phi_{n,j}[\alpha, \beta]$ if, and only if, $\mathcal{A} \models \phi'_{n,(j,\beta_j(\vec{\mu}_j^n))}[\alpha]$. It follows from the definition of γ and γ' that for all $(\vec{a}, \vec{c}) \in A^{|\vec{x}_j^n|} \times [n]_0^{|\vec{\mu}_j^n|}$ we have $(\vec{a}, \vec{c}) \in (V_j^n)^{(\mathcal{A},\gamma)}$ if, and only if, $\vec{a} \in (V_{j,\vec{c}}^n)^{(\mathcal{A},\gamma')}$. In other words γ and γ' satisfy condition 3 in the definition of a translation. We note that γ maps each second-order variable V_j^n to the flattening of Θ_n at j

and γ' maps each second-order variable $V^n_{j,\vec{c}}$ to the flattening of Θ'_n at $(j, \vec{c})$. It follows that

$$\mathcal{A} \models \phi_{n,i}[\alpha, \beta] \iff \mathcal{A} \models \theta_{n,i}[\alpha, \beta, \gamma] \iff \mathcal{A} \models \theta_{n,i;\beta}[\alpha, \gamma'] \iff \mathcal{A} \models \phi'_{n,(i,\vec{c})}[\alpha].$$

The first and third equivalences follow from the definition of a flattening and the second equivalence follows from the definition of a translation for n. The proof of Claim 3.26.1 follows by induction.

□

We have for each $n \in \mathbb{N}$ that $\Theta_n(1)$ has no free number variables. Let θ_n and θ'_n be the flattenings of Θ_n and Θ'_n. It follows that no free number variables appear in θ_n and θ'_n. Then, from Claim 3.26.1, we have that $\mathcal{A} \models \theta_n[\alpha]$ if, and only if, $\mathcal{A} \models \theta'_n[\alpha]$ for each $\mathcal{A} \in \mathrm{fin}[\rho, n]$ and assignment $\alpha \in A^{\vec{x}^n_1}$. In other words Θ' and Θ defines the same query.

Since Θ is P-uniform, the function that maps n to Θ_n is computable in time polynomial in n. Since Θ has constant length it follows that there is a constant k such that for each $n \in \mathbb{N}$ and $i \in M_n$, $|\Theta_n(i)| \leq k$. It follows from Lemma 3.25 that the function that maps each $\Theta_n(i)$ (for $n \in \mathbb{N}$, $i \in M_n$) to the translation of $\Theta_n(i)$ for n is computable in time polynomial in n. Therefore the function $n \mapsto \Theta'_n$ is computable in time polynomial in n. From Lemma 3.25 and the fact that Θ has constant length, and hence constant width, that Θ' has constant width. In summary, we have that Θ' is a P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-substitution programs with constant width and Θ' defines the same query as Θ. The result follows. □

3.5 Infinitary Logics

We now complete the translation from fixed-point logics with generalised operators to bounded-variable infinitary logics with extended quantifiers. We first note that formally we restricted our attention to finitary logics in this chapter and only defined extensions by many-sorted quantifiers for these finitary logics. However, it is easy to see that we can extend an infinitary logic by a many-sorted quantifier using essentially the same approach used to extend an infinitary logic by a Lindström quantifier.

Proposition 3.27. *Let $\mathbf{\Omega}$ be a family of almost relational operators and let $\mathbf{Q}$ be the corresponding set of quantifiers. Let ρ be a relational vocabulary. Every query definable in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$ can be defined by a formula of the form*

$$\phi(\vec{x}) \equiv \bigvee_{n \in \mathbb{N}} (\exists^{=n} x\,(x = x)) \wedge \phi_n(\vec{x}),$$

where for all $n \in \mathbb{N}$ we have $\phi_n \in \mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$. It follows that $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}) \leq \mathcal{C}^\omega(\widetilde{\mathbf{Q}})$.

Proof. Let $\theta(\vec{x}) \in \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$. It follows from Lemma 3.16 that $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}) \equiv \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}^B)$. From Lemma 3.24 there exists a P-uniform family $\mathrm{FO}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}^B)[\rho]$-substitution programs $(\Theta_n(\vec{x}))_{n \in \mathbb{N}}$

with constant length that defines the same query as θ. From Lemma 3.26 there exists a P-uniform family of FO($\widetilde{\mathbf{Q}}$)[ρ]-substitution programs $(\Phi_n(\vec{x}))_{n\in\mathbb{N}}$ with constant width that defines the same query as $(\Theta_n)_{n\in\mathbb{N}}$. For each $n \in \mathbb{N}$ let $\phi_n(\vec{x})$ be the flattening of Φ_n. It follows that for each $n \in \mathbb{N}$, ϕ_n defines the same query as θ for structures of size n, and θ and $\phi \equiv \bigwedge_{n\in\mathbb{N}}(\exists^{=n}x\,(x = x)) \wedge \phi_n(\vec{x})$ define the same query. The result follows. □

Chapter 4

Symmetric Circuits

A Boolean circuit C computing a function from $\{0,1\}^n$ to $\{0,1\}$ is usually taken (e.g. [4, 43]) to be a directed acyclic graph in which each gate g that has m incoming edges is labelled with a Boolean function $F_g : \{0,1\}^m \to \{0,1\}$ from a basis $\mathbb{B}$ and with exactly n input gates of C labelled by variables. Notice that if we allow the function F_g to be arbitrary then an order would need to be imposed on the children of g in order to ensure an unambiguous evaluation of g. When we say that C is a directed acyclic graph, without further ordering the nodes, we implicitly assume that the functions in the basis $\mathbb{B}$ are *symmetric*. That is to say F_g is invariant under all permutations of its m inputs. This unstated assumption is pervasive in circuit complexity (see [4, 43, 8]). In particular, the standard Boolean basis of AND, OR and NOT gates as well as bases with majority or threshold gates only contain symmetric functions.

We are interested in Boolean circuits that compute queries on relational structures. In the case of (undirected, loop-free) graphs on n vertices, such a circuit might compute a function from $\{0,1\}^{\binom{n}{2}}$ to $\{0,1\}$. In this case, the function computed by the circuit is not necessarily invariant under all permutations of the inputs, but it is invariant under all permutations of the $\binom{n}{2}$ inputs induced by permutations of $[n]$. We are especially concerned with *symmetric* circuits, that is those where the permutations of $[n]$ extend to automorphisms of the circuit. Note that this use of the word "symmetric" is distinct from its use when applied to Boolean functions. For such circuits, Anderson and Dawar [3] consider P-uniform families of circuits defined over the standard Boolean basis as well the extension of the standard basis with majority functions. It is a consequence of their results that the latter are strictly more powerful than the former. In particular, families of symmetric circuits with majority functions are shown to be equivalent to the logic FPC. In contrast, we show in this chapter that adding any further symmetric functions to the basis does not allow us to extend the power of P-uniform symmetric circuits beyond that of FPC.

As we ultimately aim to generalise Anderson and Dawar's result and establish circuit characterisations for a broad range of extensions of fixed-point logic, many of which are

strictly more expressive than FPC, we need to consider circuits with gates that compute non-symmetric Boolean functions. To lead up to this, we first develop a general framework of *structured Boolean functions*. These are functions whose inputs naturally encode τ-structures. We are particularly interested in *isomorphism-invariant* structured functions, i.e. those structured functions whose output is invariant under the symmetries of these structures.

We next introduce a more general circuit model whose gates are labelled by isomorphism-invariant structured functions, rather than symmetric functions. We generalise various important notions introduced by Anderson and Dawar, including the notions of a circuit automorphism and a symmetric circuit. We aim to replicate many of Anderson and Dawar's [3] results in this more general setting. However, many of the proof techniques used in their paper heavily rely, often in quite subtle ways, on the assumption that individual gates compute symmetric functions. In this chapter we discuss some difficulties that arise in this more general setting and what implications these have for the prospect of generalising some of Anderson and Dawar's results.

This chapter is organised as follows. In Section 4.1 we introduce the theory of structured Boolean functions. In Section 4.2 we introduce our more general circuit model and correspondingly generalise key notions for such circuits. We also discuss some new concepts which only become relevant in this more general setting and which will be needed to overcome some of the difficulties introduced by this generalisation. We also formally state the main theorem of this thesis. In Section 4.3 we show that any query definable by a P-uniform family of symmetric circuits over a basis of symmetric functions is definable by a P-uniform family of symmetric circuits over the standard basis with majority functions.

4.1 Structured Functions and Symmetry

In complexity theory we are often interested in the problem of deciding a language $L \subseteq \{0,1\}^*$. We can equivalently define L using a family of functions $(F_n : \{0,1\}^n \to \{0,1\})_{n\in\mathbb{N}}$, where each F_n decides L for strings of length n. In contrast, in descriptive complexity we are usually interested in working directly with classes of structures, rather than sets of string encodings, and consider methods of computation that respect the abstract symmetry properties of these structures. In this context we are interested in Boolean functions that take as input a more direct encoding of a structure and which respect the appropriate symmetries. We do this by considering families of functions of the form $F : \{0,1\}^X \to \{0,1\}$ where X is not just an (ordered) index set, such as $[n]$, but carries with it a structure which gives a natural way of identifying input functions with relational structures. To give an example, for a finite set V if we take $X = V \times V$ then the functions $f : X \to \{0,1\}$ can each be identified with a directed graph with vertex set V. The function F can then be thought of as deciding a set of directed graphs with vertex set V. When the index set X is unstructured it is natural to consider those Boolean functions invariant under *all* permutations in $\mathbf{Sym}_X$ on the input, i.e. the

symmetric functions. However, in our example a more natural symmetry requirement would be to consider functions invariant under the action of $\mathbf{Sym}_V$ on the input. Understanding the inputs to this function as directed graphs, this notion of invariance corresponds to the function being invariant under graph isomorphism. We use this example to motivate our introduction of *structured functions*.

Definition 4.1. Let $\tau := (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary. Let $X = \uplus_{s \in \boldsymbol{S}} X_s$ be a disjoint union of non-empty sets. Let $K_{\tau,X}$ be the complete τ-structure with universe X. Let $\tau[X] := \uplus_{R \in \boldsymbol{R}} R^{K_{\tau,X}} = \{(\vec{a}, R) : R \in \boldsymbol{R}, \vec{a} \in R^{K_{\tau,X}}\}$. If $X = [n]$ for some $n \in \mathbb{N}$ we write $\tau[n]$ to denote $\tau[X]$.

We call a function $F : \{0,1\}^{\tau[X]} \to \{0,1\}$ a *structured function* with *universe* τ and *domain* X. We call $K_{\tau,X}$ the *structure* associated with F and $\tau[X]$ the *index set* of F. We denote the index set of F by $\mathrm{ind}(F)$.

We always assume that if F is a structured function with vocabulary $\tau = (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ then for each $s \in \boldsymbol{S}$ there exists $R \in \boldsymbol{R}$ such that s appears in $\zeta(R)$. Let $\tau = (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary and let $X := \uplus_{s \in \boldsymbol{S}} X_s$ be a disjoint union of non-empty sets. We identify each $f : \tau[X] \to \{0,1\}$ with the τ-structure $\mathcal{A}$ with universe X and such that for each $R \in \boldsymbol{R}$, $R^{\mathcal{A}} = \{\vec{a} : f(\vec{a}, R) = 1\}$.

There is a group action of $\mathbf{Sym}_{\tau[X]}$ on $\{0,1\}^{\tau[X]}$ defined for each $\sigma \in \mathbf{Sym}_{\tau[X]}$ and $f \in \{0,1\}^{\tau[X]}$ such that $(f \cdot \sigma)(x) = f(\sigma(x))$ for all $x \in \tau[X]$. There is also a group action of $\mathbf{Aut}(K_{\tau,X})$ on $\{0,1\}^{\tau[X]}$ defined for each $\lambda \in \mathbf{Aut}(K_{\tau,X})$ and $f \in \{0,1\}^{\tau[X]}$ such that $(f \cdot \lambda)(\vec{a}, R) = (\lambda(\vec{a}), R)$ for all $(\vec{a}, R) \in \tau[X]$. We can identify $\mathbf{Aut}(K_{\tau,X})$ with a subgroup of $\mathbf{Sym}_{\tau[X]}$. We notice that for all $\sigma \in \mathbf{Sym}_{\tau[X]}$ we have for $f : \tau[X] \to \{0,1\}$ that $f \cdot \sigma$ is isomorphic to f if, and only if, $\sigma \in \mathbf{Aut}(K_{\tau,X})$. We call $\mathbf{Aut}(K_{\tau,X})$ *the automorphism group of* F.

Definition 4.2. Let $F : \{0,1\}^{\tau[X]} \to \{0,1\}$ be a structured function. Let $G \leq \mathbf{Sym}_{\tau[X]}$. We say that F is *G-invariant* if for all $\sigma \in G$ and all $f : \tau[X] \to \{0,1\}$ we have $F(f) = F(\sigma f)$. We say that F is *isomorphism-invariant* if F is $\mathbf{Aut}(K_{\tau,X})$-invariant.

It follows from the above argument that F is an isomorphism-invariant if, and only if, F decides a property of τ-structures. We have noted that each function $f : \tau[X] \to \{0,1\}$ can be identified with a τ-structure with universe X. Let H be a set. We can similarly identify a function $f : \tau[X] \to H$ with an (edge) labelled version of the complete τ-structure with universe X. We say two labelled structures are isomorphic if there is a bijection between these structures that preserves relations and labels. In other words, if $f : \tau[X] \to H$ and $g : \tau[Y] \to H$ then f and g are isomorphic if, and only if, there exists $\pi : X \to Y$ such that for all $(\vec{a}, R) \in \tau[X]$, $f(\vec{a}, R) = g(\pi(\vec{a}), R)$. If f and g have the same domain then we can identify this bijection with an element of $\mathbf{Aut}(K_{\tau,X})$.

We now return to the motivating example mentioned earlier involving Boolean functions that take directed graphs as inputs. We notice that for a finite set V the Boolean function

$F : \{0,1\}^{V \times V} \to \{0,1\}$ can be viewed as a structured function with universe V and with vocabulary τ, where τ is single-sorted and contains a single binary relation symbol. We notice that F is isomorphism-invariant as a structured function if, and only if, $F(\mathcal{A}) = F(\mathcal{B})$ whenever $\mathcal{A}$ and $\mathcal{B}$ are isomorphic directed graphs.

We can similarly view a Boolean function of the form $F : \{0,1\}^n \to \{0,1\}$ as a structured function with universe $[n]$ and vocabulary τ, where τ is single-sorted and contains a single unary relation. In this case we notice that F is isomorphism-invariant if, and only if, F is invariant under arbitrary permutations of the input string. In this way we recover the usual notions of a Boolean function and a symmetric Boolean function. We identify the usual symmetric Boolean functions with isomorphism-invariant structured functions defined over a single-sorted vocabulary with a single unary relation. It would be natural then to call isomorphism-invariant structured functions that are not symmetric non-symmetric Boolean functions. However, this could be confusing as these functions are invariant with respect to the automorphism group of the relevant complete structure, and so are "symmetric" in some appropriately weaker sense. As such, in our framework we call symmetric functions *trivially automorphism-invariant* (or just *trivially invariant*) functions and say that isomorphism-invariant structured function that are not trivially automorphism-invariant are *non-trivially automorphism-invariant* or just *non-trivially invariant*.

We can also define a structured function that takes 0-1 matrices as inputs. In this case we take $\tau = (\{M\}, \{s_1, s_2\}, \zeta)$ where $\zeta(M) = (s_1, s_2)$. We can think of a τ-structure as being a 0-1 matrix with the first sort denoting the row index set and the second sort denoting the column index set. In this case a structured function $F : \{0,1\}^{\tau[X]} \to \{0,1\}$ is isomorphism-invariant if, and only if, it is invariant under the action of all permutations of the form (σ_1, σ_2) where σ_1 permutes the first sort (thought of as the row indices) and σ_2 permutes the second sort (thought of as the column indices). In other words, F is isomorphism-invariant if, and only if, it is invariant under all row-column permutations of the matrix. We now generalise the usual notion of a Boolean basis so as to allow for structured functions.

Definition 4.3. A *Boolean basis* (or just a *basis*) is a set of isomorphism-invariant structured functions.

We now introduce some notation and a few useful conventions. If F is a structured function with a vocabulary τ consisting of a single relation symbol we often omit it when denoting the index set. In other words, if τ has a single relation symbol U we write $a \in \tau[X]$ instead of $(a, U) \in \tau[X]$. It is conventional when working with Boolean functions to take the index set to be an initial segment of the natural numbers. We also follow this convention and often work with structured functions where each sort in its universe is an initial segment of the natural numbers. Let $\tau = (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be a many-sorted relational vocabulary. Let $e : \boldsymbol{S} \to \mathbb{N}$. We write $\tau[e]$ to denote $\tau[X]$ where $X = \uplus_{s \in \boldsymbol{S}} X_s$ and $X_s = [e(s)]$ for each $s \in \boldsymbol{S}$. In some contexts it is natural to consider the set of sort symbols to be ordered.

Suppose $\boldsymbol{S} = \{s_1, \ldots, s_m\}$. Then for $p_1, \ldots, p_m \in \mathbb{N}$ we write $\tau[p_1, \ldots, p_m]$ to denote $\tau[e]$ where $e : \boldsymbol{S} \to \mathbb{N}$ is defined by $e(s_i) = p_i$ for each $i \in [m]$.

We noted earlier that a language is defined by a family of Boolean functions. We now show that a class of structures can similarly be associated with a family of structured functions. Let $\mathcal{G}$ be a class of τ-structures and let $X = \uplus_{s \in \boldsymbol{S}} X_s$ be a disjoint union of non-empty sets. Let $F_{\mathcal{G}}[X] : \{0,1\}^{\tau[X]} \to \mathbb{N}$ be defined by $F_{\mathcal{G}}[X](\mathcal{A}) = \mathrm{id}_{\mathcal{G}}(\mathcal{A})$ for all $\mathcal{A} \in \mathrm{fin}[\tau, X]$. Let $\mathbb{B}_{\mathcal{G}}$ be the set of all $F_{\mathcal{G}}[X]$ for any union of non-empty disjoint sets $X = \uplus_{s \in \boldsymbol{S}} X_s$. We call $\mathbb{B}_{\mathcal{G}}$ *the basis corresponding to* $\mathcal{G}$. Notice that each structured function in $\mathbb{B}_{\mathcal{G}}$ is isomorphism-invariant. We say that a basis $\mathbb{B}$ is *finitely generated* if there is a finite collection of classes of structures $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ such that $\mathbb{B} = \bigcup_{i \in [k]} \mathcal{G}_i$.

We also introduce some notation for structured functions defined over universes consisting of disjoint unions of initial segments of the natural numbers. If $\boldsymbol{S} = \{s_1, \ldots, s_m\}$ and $p_1, \ldots, p_m \in \mathbb{N}$ we write $F_{\mathcal{G}}[p_1, \ldots, p_m]$ to denote $F_{\mathcal{G}}[\uplus_{s_i \in \boldsymbol{S}}[p_i]]$.

We can also associate each family of almost relational generalised operators with a basis of isomorphism-invariant structured functions. We notice that each Boolean-valued almost relational generalised operator Ω and each assignment α to the constant symbols in the vocabulary of Ω defines a class of many-sorted relational structures $\mathcal{G}_\alpha$. The *basis corresponding to* Ω is the union of the bases $\mathbb{B}_{\mathcal{G}_\alpha}$ for each assignment α. We now define this notion formally.

Definition 4.4. Let Ω be an almost relational generalised operator. Let Ω^B be the corresponding Boolean-valued generalised operator. Let E be the evaluation function of Ω^B. Let $\tau := (\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{F}, \zeta)$ be the vocabulary of Ω^B. Let $\tau_{\mathbf{rel}} := (\boldsymbol{R}, \boldsymbol{S}, \zeta_{\mathbf{rel}})$ where $\zeta_{\mathbf{rel}} = \zeta\big|_{\boldsymbol{R}}$. For each $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ and $\mathcal{A} \in \mathrm{fin}[\tau_{\mathbf{rel}}]$ let $(\mathcal{A}|\alpha)$ be the τ-structure $(A, (R^{\mathcal{A}})_{R \in \boldsymbol{R}}, (\alpha(F))_{F \in \boldsymbol{F}})$. For each $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ let $\mathcal{G}_{E,\alpha} := \{\mathcal{A} \in \tau_{\mathbf{rel}} : E(\mathcal{A}|\alpha) = 1\}$. Let $\mathbb{B}_\Omega := \{\mathbb{B}_{\mathcal{G}_{E,\alpha}} : \alpha : \boldsymbol{F} \to \mathbb{N}_0\}$. In order to simplify notation we write $F_{\Omega,\alpha}[X]$ to denote $F_{\mathcal{G}_{E,\alpha}}[X]$ for each $\tau_{\mathbf{rel}}$-sorted set X. We call $\mathbb{B}_\Omega$ the *basis corresponding to* Ω. Let $\boldsymbol{\Omega}$ be a set of almost relational generalised operators. Let $\mathbb{B}_{\boldsymbol{\Omega}} := \bigcup_{\Omega \in \boldsymbol{\Omega}} \mathbb{B}_\Omega$. We call $\mathbb{B}_{\boldsymbol{\Omega}}$ *the basis corresponding to* $\boldsymbol{\Omega}$. If Ω is relational then $\tau_{\mathbf{rel}} = \tau$ and we omit α from the above subscripts and write $\mathcal{G}_E$ to denote $\mathcal{G}_{E,\alpha}$ and $F_\Omega[X]$ to denote $F_{\mathcal{G}_E}[X]$ for each τ-sorted set X.

We introduce some specific notation for vectorised operators. We first note that if $\boldsymbol{\Omega}$ is an almost relational vectorised operator then for any $\Omega \in \boldsymbol{\Omega}$ we have $\mathbb{B}_{\boldsymbol{\Omega}} = \mathbb{B}_\Omega$. In this case we write $F_{\boldsymbol{\Omega},\alpha}[X]$ to denote $F_{\Omega,\alpha}[X]$ for each $\Omega \in \boldsymbol{\Omega}$. We sometimes use $\boldsymbol{\Omega}$ to denote a set of *vectorised operators*, rather than a set of generalised operators. In this case we abuse notation and write $\mathbb{B}_{\boldsymbol{\Omega}}$ to denote the union of $\mathbb{B}_{\boldsymbol{\Omega}_i}$ for each vectorised operator $\boldsymbol{\Omega}_i \in \boldsymbol{\Omega}$.

We now consider the counting operator as an example and show that the corresponding Boolean basis is the set of threshold functions.

Example 4.5. Let $\boldsymbol{\Omega}_{E_{\mathbf{cnt}}}$ be the counting operator. Let s be the single sort symbol in the vocabulary of $\boldsymbol{\Omega}_{E_{\mathbf{cnt}}}$. Then $\mathbb{B}_{\boldsymbol{\Omega}_{E_{\mathbf{cnt}}}}$ consists of all functions of the form $F_{\boldsymbol{\Omega}_{E_{\mathbf{cnt}}},\alpha}[X]$ where

$\alpha : \{s\} \to \mathbb{N}_0$ and $X : \{s\} \to \mathbb{N}_0$ and $F_{\mathbf{\Omega},\alpha}[X]$ takes as input the characteristic function of a subset of $[X(s)]$ and outputs 1 if, and only if, the size of that set is at least $\alpha(s)$.

4.2 Symmetric Circuits

We now generalise the circuit model of Anderson and Dawar [3] so as to allow for circuits to be defined over bases that may include both trivially and non-trivially invariant functions. In this model each gate g is not only associated with an element of the basis, as in the conventional case, but also with a labelling function. This labelling function maps the input gates of g to an appropriate set of labels (i.e. the index of the structured function associated with g). In concord with this generalisation, we also update the circuit-related notions discussed by Anderson and Dawar [3], e.g. circuit automorphisms, symmetry, etc. Moreover, we briefly discuss some of the important complications introduced by our generalisation, and introduce some of the important tools we will use in later chapters to address these complications.

Definition 4.6 (Circuits on Structures)**.** Let $\mathbb{B}$ be a basis and ρ be a relational vocabulary, we define a $(\mathbb{B}, \rho)$-*circuit* C of order n computing a q-ary query Q as a structure $\langle G, \Omega, \Sigma, \Lambda, L\rangle$.

- G is called the set of gates of C.
- Ω is an injective function from $[n]^q$ to G. The gates in the image of Ω are called the output gates. When $q = 0$, Ω is a constant function mapping to a single output gate.
- Σ is a function from G to $\mathbb{B} \uplus \rho \uplus \{0, 1\}$ such that $|\Sigma^{-1}(0)| \leq 1$ and $|\Sigma^{-1}(1)| \leq 1$. Those gates mapped to $\rho \uplus \{0, 1\}$ are called input gates, with those mapped to ρ called relational gates and those mapped to $\{0, 1\}$ called constant gates. Those gates mapped to $\mathbb{B}$ are called internal gates.
- Λ is a sequence of injective functions $(\Lambda_R)_{R\in\rho}$ such that Λ_R maps each relational gate g with $\Sigma(g) = R$ to the tuple $\Lambda_R(g) \in [n]^{r_R}$. When no ambiguity arises we write $\Lambda(g)$ for $\Lambda_R(g)$.
- L associates with each internal gate g a function $L(g) : \mathrm{ind}(\Sigma(g)) \to G$ such that if we define a relation $W \subseteq G^2$ by $W(h_1, h_2)$ iff h_2 is an internal gate and h_1 is in the image of $L(h_2)$, then (G, W) is a directed acyclic graph.

The definition requires some explanation. Each gate in G computes a function of its inputs and the relation W on G is the set of "wires". That is, $W(h, g)$ indicates that the value computed at h is an input to g. However, since the functions are structured, we need more information on the set of inputs to g and this is provided by the labelling L. $\Sigma(g)$ tells us what the function computed at g is, and thus the index of $\Sigma(g)$ tells us the structure on the inputs and $L(g)$ maps this to the set of gates that form the inputs to g.

Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit of order n. Recall that the order of a circuit refers to the size of the input structures it takes. In contrast, we define the *size* of C, denoted

$|C|$ to be the number of elements in G. If $(C_n)_{n\in\mathbb{N}}$ is a family of circuits we assume that each C_n is a circuit of order n.

For each $g \in G$ we let $W(\cdot, g) := \{h \in G : W(h, g)\}$ and $W(g, \cdot) := \{h \in G : W(g, h)\}$. We call the elements of $W(\cdot, g)$ the *children* of g and the elements of $W(g, \cdot)$ the *parents* of g. We also abbreviate $W(g, \cdot)$ by H_g. We write W_T for the transitive closure of W.

For a gate $g \in G$ with $\Sigma(g) \in \mathbb{B}$, we let the *index* of g, denoted by $\mathrm{ind}(g)$, be the index of $\Sigma(g)$. We let the vocabulary and universe of g be the vocabulary and universe of $\Sigma(g)$, respectively, and denote the vocabulary of g by $\mathrm{voc}(g)$ and the universe by $\mathrm{unv}(g)$. We let the automorphism group of g be the automorphism group of $\Sigma(g)$ and we write $\mathbf{Aut}(g)$ to denote this group. We write $\mathrm{str}(g)$ denote the complete structure associated with $\Sigma(g)$. We say that a gate g is *trivially automorphism-invariant* (or just *trivially invariant*)if $\Sigma(g)$ is a trivially invariant function and otherwise we say that g is *non-trivially automorphism invariant* (or just *non-trivially invariant*. We say C is a *circuit with trivially invariant gates* if every gate in C is trivially automorphism-invariant.

Let ρ be a relational vocabulary, $\mathcal{A}$ be a ρ-structure with universe A of size n, and $\gamma \in [n]^{\underline{A}}$. Let $\gamma\mathcal{A}$ be the structure with universe $[n]$ formed by mapping the elements of A in accordance with γ. The evaluation of a $(\mathbb{B}, \rho)$-circuit C of order n computing a q-ary query Q proceeds by recursively evaluating the gates in the circuit. The evaluation of the gate g for the bijection γ and input structure $\mathcal{A}$ is denoted by $C[\gamma\mathcal{A}](g)$, and is given as follows

1. If g is a constant gate then it evaluates to the bit given by $\Sigma(g)$,
2. if g is a relational gate then g evaluates to true iff $\gamma\mathcal{A} \models \Sigma(g)(\Lambda(g))$, and
3. if g is an internal gate let $L^{\gamma\mathcal{A}}(g) : \mathrm{ind}(g) \to \{0, 1\}$ be defined by $L^{\gamma\mathcal{A}}(g)(x) = C[\gamma\mathcal{A}](L(g)(x))$, for all $x \in \mathrm{ind}(g)$. Then g evaluates to true if, and only if, $\Sigma(g)(L^{\gamma}_g) = 1$.

We say that C defines the q-ary query $Q \subseteq A^q$ under γ where $\vec{a} \in Q$ if, and only if, $C[\gamma\mathcal{A}](\Omega(\gamma\vec{a})) = 1$. We write $C[\gamma\mathcal{A}]$ to denote the query Q.

We notice that in general the output of a circuit C may depend on the particular encoding of $\mathcal{A}$ as a structure with universe $[n]$, i.e. on the chosen bijection γ. It is natural to consider circuits whose outputs do do not depend on this choice of encoding. Anderson and Dawar [3] call such a circuit *invariant*. We have reproduced their definition below.

Definition 4.7 (Invariant Circuit)**.** Let C be a $(\mathbb{B}, \rho)$-circuit of order n, computing some q-ary query. We say C is *invariant* if for every ρ-structure $\mathcal{A}$ of size n, $\vec{a} \in A^q$, and $\gamma_1, \gamma_2 \in [n]^{\underline{A}}$ we have that $C[\gamma_1\mathcal{A}](\Omega(\gamma_1\vec{a})) = C[\gamma_2\mathcal{A}](\Omega(\gamma_2\vec{a}))$.

If a family of $(\mathcal{B}, \rho)$-circuits $\mathcal{C}$ is invariant it follows that the query computed is a q-ary query on ρ-structures. Thus if $q = 0$ then $\mathcal{C}$ computes a property of ρ-structures. The following lemma allows us to recast this notion in terms of the language developed in this thesis.

Lemma 4.8. *Let C be a $(\mathbb{B}, \rho)$-circuit of order n computing a 0-ary query. The function computed by C is isomorphism-invariant if, and only if, C is an invariant circuit.*

Proof. We first note that, since C computes a 0-ary query, there is exactly one output gate. We call this gate g_o. We can associate with each relational gate g in C a unique pair $(\vec{a}, R)$, where $R := \Sigma(g)$ is a relational symbol in τ and $\vec{a} := \Lambda_R(g)$ is an an element of $R^{[n]}$. Thus we can think of the function F_C computed by the circuit as having its input string indexed by the elements of $\rho[n]$, and so think of F_C as a structured function with index $\rho[n]$.

We now present a number of observations, and then combine these observations to prove both directions of the lemma. Let $\mathcal{B}$ be a ρ-structure of size n over the universe $[n]$. We can define a function $f_\mathcal{B} : \rho[n] \to \{0,1\}$ such that $f_\mathcal{B}(\vec{a}_R) = 1$ if, and only if, $\vec{a} \in R^\mathcal{B}$. Let $f : \rho[n] \to \{0,1\}$. We can define the ρ-structure $\mathcal{B}_f$ over the universe $[n]$ such that for all $R \in \rho$, we have $\vec{a} \in R^{\mathcal{B}_f}$ if, and only if, $f(\vec{a}_R) = 1$. It is easy to see that the functions $\mathcal{B} \mapsto f_\mathcal{B}$ and $f \mapsto \mathcal{B}_f$ are inverse to one another, and so define a bijection. Moreover, we notice that for all $\sigma \in \mathbf{Sym}_n$ we have that $f_{\sigma\mathcal{B}}(\vec{a}_R) = 1$ if, and only if, $\vec{a} \in R^{\sigma\mathcal{B}}$ if, and only if, $\sigma^{-1}\vec{a} \in R^\mathcal{B}$ if, and only if, $(f_\mathcal{B}\sigma^{-1})(\vec{a}_R) = f_\mathcal{B}(\sigma^{-1}\vec{a}_R) = 1$. It follows $f_{\sigma\mathcal{B}} = f_\mathcal{B}\sigma^{-1}$.

Let $\mathcal{A}$ be a ρ-structure of size n with universe A, and let $\gamma_1, \gamma_2 \in [n]^{\underline{A}}$. From the definition of F_C it follows that for $\gamma \in \mathbf{Sym}_n$ we have that $F_C(f_{\gamma\mathcal{A}}) = 1$ if, and only if, $C[\gamma\mathcal{A}](g_o) = 1$.

Let $\gamma_1, \gamma_2 \in [n]^{\underline{A}}$ and let $\sigma \in \mathbf{Sym}_n$ be such that $\gamma_1 = \sigma\gamma_2$. We have that

$$\begin{aligned} C[\gamma_1\mathcal{A}](g_o) = C[\gamma_2\mathcal{A}](g_o) &\Leftrightarrow F_C(f_{\gamma_1\mathcal{A}}) = F_C(f_{\gamma_2\mathcal{A}}) \\ &\Leftrightarrow F_C(f_{\gamma_1\mathcal{A}}) = F_C(f_{\sigma^{-1}\gamma_1\mathcal{A}}) \\ &\Leftrightarrow F_C(f_{\gamma_1\mathcal{A}}) = F_C(f_{\gamma_1\mathcal{A}\sigma}). \end{aligned}$$

We now combine the above observations to prove the result. Suppose F_C is isomorphism-invariant. Let $\gamma_1, \gamma_2 \in [n]^{\underline{A}}$ and let $\sigma \in \mathbf{Sym}_n$ be such that $\gamma_1 = \sigma\gamma_2$. Then, from isomorphism-invariance, we have $F_C(f_{\gamma_1\mathcal{A}}) = F_C(f_{\gamma_1\mathcal{A}\sigma})$, and so $C[\gamma_1\mathcal{A}](g_o) = C[\gamma_2\mathcal{A}](g_o)$.

Suppose C is invariant. Fix a bijection $\gamma_1 \in [n]^{\underline{A}}$. Let $\sigma \in \mathbf{Sym}_n$ and $f : \rho[n] \to \{0,1\}$. Let $\mathcal{A} = \gamma_1^{-1}\mathcal{B}_f$ and $\gamma_2 := \sigma^{-1}\gamma_1$. Since C is invariant we have $C[\gamma_1\mathcal{A}](g_o) = C[\gamma_2\mathcal{A}](g_o)$, and so $F_C(f_{\gamma_1\mathcal{A}\sigma}) = F_C(f_{\gamma_1\mathcal{A}})$. But $f_{\gamma_1\mathcal{A}} = f_{\mathcal{B}_f} = f$, and so $F_C(f\sigma) = F_C(f)$. It follows that F_C is isomorphism-invariant.

□

We now define an automorphism of a circuit, generalising the definition introduced by Anderson and Dawar. The definition is similar, but adds the requirement that if a gate g is mapped to g', then children of g must be mapped to the children of g' via some appropriate isomorphism of the structure associated with g.

Definition 4.9 (Automorphism)**.** Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \tau)$-circuit of order n computing a q-ary query, and where $\mathbb{B}$ is a basis of isomorphism-invariant structured functions. Let $\sigma \in \mathbf{Sym}_n$ and $\pi : G \to G$ be a bijection such that

- for all output tuples $x \in [n]^q$, $\pi\Omega(x) = \Omega(\sigma x)$,
- for all gates $g \in G$, $\Sigma(g) = \Sigma(\pi g)$,
- for each relational gate $g \in G$, $\sigma\Lambda(g) = \Lambda(\pi g)$, and
- For each pair of gates $g, h \in G$ $W(h, g)$ if and only if $W(\pi h, \pi g)$ and for each internal gate g we have that $L(\pi g)$ and $\pi L(g)$ are isomorphic (as labelled structures).

We call π an *automorphism* of C, and we say that σ *extends to an automorphism* π. The group of automorphisms of C is called $\mathbf{Aut}(C)$.

We can equally define an isomorphism between a pair of circuits $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ and $C' = \langle G', \Omega', \Sigma', \Lambda', L'\rangle$ as a bijection $\pi : G \to G'$ satisfying conditions as above. We do not usually need to consider distinct, isomorphic circuits and for this reason we only formally define automorphsims.

We are particularly interested in circuits that have the property that *every* permutation in $\mathbf{Sym}_n$ extends to an automorphism of the circuit.

Definition 4.10 (Symmetry)**.** A circuit C or order n is called *symmetric* if every $\sigma \in \mathbf{Sym}_n$ extends to an automorphism on C.

It follows that for any symmetric circuit C of order n there is a homomorphism h that maps $\mathbf{Sym}_n$ to $\mathbf{Aut}(C)$ such that if $\sigma \in \mathbf{Sym}_n$ then $h(\sigma)$ is an an automorphism extending σ. Suppose C does not contain a relational gate labelled by a relation symbol with non-zero arity. In that case C computes a constant function. For this reason, in this thesis we always assume a circuit contains at least one relational gate with non-zero arity. Now, by assumption there exists a relational gate in C such that some element of $[n]$ appears in the tuple labelling that gate. By symmetry it follows that every element of $[n]$ appears in a tuple labelling a relational gate in C. It follows that no two distinct elements of $\mathbf{Sym}_n$ agree on all input gates, and so the homomorphism h is injective.

If this homomorphism is also surjective then we have that each element of σ extends uniquely to an automorphism of the circuit. In this case we say that a circuit has *unique extensions*.

Definition 4.11. We say that a circuit C has *unique extensions* if for every $\sigma \in \mathbf{Sym}_n$ there is at most one $\pi_\sigma \in \mathbf{Aut}(C)$ such that π_σ extends σ.

It is worth noting that many of the important technical tools needed in this thesis are only applicable if the circuit under consideration has unique extensions. In order to handle this technicality Anderson and Dawar [3] introduce the notion of a *rigid* circuit and show that, for circuits defined over a basis of trivially invariant functions there is a polynomial-time algorithm that takes as input a symmetric circuit and outputs an equivalent rigid symmetric circuit. They also show that a great many properties of rigid circuits can be decided in polynomial time, a necessary step in their argument, and, importantly, that rigid circuits

have unique extensions. This allows them to restrict their attention to P-uniform families of symmetric circuits (over bases of trivially invariant functions) *with unique extensions*, without a loss of generality.

In order to make use of these technical tools, as well as ensure the polynomial-time decidability of many important circuit properties, we should like to be able to construct a normal form analogous to rigidity and present a polynomial time algorithm that transforms a circuit into an equivalent circuit of this form. It is at this point that we arrive at the first complication introduced by our generalisation. The polynomial-time translation in [3] makes indispensable use of the polynomial-time decidability of many important circuit properties for circuits defined over trivially invariant bases. However, for the more general circuits discussed here, it is not known if even the most basic circuit properties are polynomial-time decidable. This is essentially due to the requirement built in to Definition 4.9 that an automorphism that takes g to g' must be an isomorphism between $L(g)$ and $L(g')$, which makes checking the condition as hard as isomorphism checking. Indeed, we show in Chapter 7 that for circuits that include gates computing non-trivially invariant functions all of the relevant circuit properties of interest are at least as hard to decide as the graph-isomorphism problem. As such, constructing an argument analogous to [3], as well as establishing the numerous other crucial results whose proofs rely on the polynomial-time decidability of various circuit properties, would be beyond the scope of this thesis.

In order to proceed we explicitly restrict our attention to a particular class of circuits characterised by a restriction on the children of gates labelled by non-trivially invariant functions. We say such circuits are *transparent*. We show in Chapter 7 that all of the circuit properties of interest are polynomial-time decidable for transparent circuits, and we use these results to define a polynomial-time transformation from transparent circuits to equivalent circuits with unique extensions. Importantly, while the restriction to transparent circuits makes it easier to translate families of circuits into formulas, the usual translation from formulas to circuits does not produce a family of transparent circuits. We thus define a novel translation from formulas to circuits in Chapter 5.

Before we can formally define *transparency* we need to define the *syntactic-equivalence* relation on the gates of a circuit. The intuition is that two gates g and g' in a circuit are syntactically-equivalent if the circuits underneath them are 'hereditarily equivalent', i.e. if the two circuits induced by the restrictions to $W_T(\cdot, g)$ and $W_T(\cdot, g')$ are in some sense just copies of one another.

Definition 4.12. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit of order n. We recursively define the equivalence relation *syntactic-equivalence*, which we denote using the symbol '$\equiv$', on G as follows. Suppose g and h are gates in C such that $\Sigma(g) = \Sigma(h)$ and either both g and h are output gates and $\Omega^{-1}(g) = \Omega^{-1}(h)$ or neither are output gates. Suppose g and h are input gates, then $g \equiv h$ if, and only if, both g and h are constant gates and $\Sigma(g) = \Sigma(h)$ or

both are relational gates and $\Lambda(g) = \Lambda(h)$. Suppose g and h are internal gates and suppose we have defined the syntactic-equivalence relation for all gates of depth less than the depth of either g or h. Then $g \equiv h$ if, and only if, $L(g)/_{\equiv}$ and $L(h)/_{\equiv}$ are isomorphic.

For a circuit C of order n and a gate g we think of g as computing the function that maps an input structure $\mathcal{A}$ and a bijection γ from the universe of $\mathcal{A}$ to $[n]$ to the evaluation $C[\gamma\mathcal{A}](g)$. While we would like to be able to identify gates that compute the same function in this sense, it is not hard to show that deciding this equivalence relation for a given circuit is NP-hard. We now show that if two gates are syntactically-equivalent then the functions computed at these two gates must be equal. We show later that the syntactic-equivalence relation is polynomial-time decidable for the class of circuits of interest to us in this thesis. In this sense we shall treat syntactic-equivalence as a tractable refinement of the NP-complete relation.

Lemma 4.13. *Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit of order n. Let $\mathcal{A}$ be a ρ-structure of size n and let γ be a bijection from the universe of $\mathcal{A}$ to $[n]$. For all $g, g' \in G$ if $g \equiv g'$ then $C[\gamma\mathcal{A}](g) = C[\gamma\mathcal{A}](g')$.*

Proof. We prove the result by induction on depth. Suppose g and g' have depth 0 and $g \equiv g'$. Then they are both input gates and so $g = g'$. The result for depth 0 then follows trivially. Suppose g and g' are internal gates, and suppose for all $h, h' \in G$ of depth less than g or g' we have that if $h \equiv h'$ then $C[\gamma\mathcal{A}](h) = C[\gamma\mathcal{A}](h')$. Suppose $g \equiv g'$. There exists $\lambda \in \mathbf{Aut}(g)$ such that $L(g)(x) \equiv L(g')(\lambda x)$ for all $x \in \mathrm{ind}(g)$. It follows from the inductive hypothesis that $L^{\gamma\mathcal{A}}(g)(x) = C[\gamma\mathcal{A}](L(g)(x)) = C[\gamma\mathcal{A}](L(g')(\lambda x)) = (L^{\gamma\mathcal{A}}(g)\lambda)(x)$ for all $x \in \mathrm{ind}(g)$. Since $\Sigma(g)$ (and so $\Sigma(g')$) is a structured function, it follows that $C[\gamma\mathcal{A}](g) = \Sigma(g)(L^{\gamma\mathcal{A}}(g)) = \Sigma(g')(L^{\gamma\mathcal{A}}(g')\lambda) = \Sigma(g')(L^{\gamma\mathcal{A}}(g')) = C[\gamma\mathcal{A}](g')$. The result follows. □

The syntactic-equivalence relation identifies gates that have 'equivalent' circuits underneath them. A similar intuition is captured by identifying gates that are mapped to one another by automorphisms of the circuit that extend the trivial permutation. We now show that if two automorphisms extend the same permutation then the two images of each gate must be syntactically-equivalent.

Lemma 4.14. *Let C be a circuit of order n, $\sigma \in \boldsymbol{Sym}_n$ and $\pi, \pi' \in \boldsymbol{Aut}(C)$ both extend σ, then for every gate g in the circuit we have that $\pi(g)$ and $\pi'(g)$ are syntactically-equivalent.*

Proof. We first note that, from the definition of an automorphism, for any gate g in C, $\Sigma(g) = \Sigma(\pi(g)) = \Sigma(\pi'(g))$, and either all of g, $\pi(g)$ and $\pi(g')$ are output gates and $\pi(g) = \pi\Omega(\Omega^{-1}(g)) = \Omega(\sigma\Omega^{-1}(g)) = \pi'\Omega(\Omega^{-1}(g)) = \pi'(g)$, or none of g, $\pi(g)$ and $\pi'(g)$ are output gates.

We now prove the result by induction on depth. Suppose g is a gate of depth 0. Then g is either a relational or constant gate. In either case $\pi(g) = \pi'(g)$, and so $\pi(g)$ and $\pi'(g)$ are syntactically-equivalent.

Suppose g is an internal gate and suppose that for every gate h of depth less than g, $\pi(h)$ is syntactically-equivalent to $\pi'(h)$. We have that there exists $\lambda, \lambda' \in \mathbf{Aut}(g)$ such that $\pi L(g)(x) = L(\pi g)(\lambda x)$ and $\pi' L(g)(x) = L(\pi' g)(\lambda' x)$, for all $x \in \mathrm{ind}(g)$. Then, from the inductive hypothesis, we have that $\pi L(g)(x) \equiv \pi' L(g)(x)$ and so $L(\pi g)(\lambda x) \equiv L(\pi' g)(\lambda' x)$, for all $x \in \mathrm{ind}(g)$. Thus we have that $L(\pi g)(x) = L(\pi g)(\lambda\lambda^{-1}(x)) \equiv L(\pi' g)(\lambda'\lambda^{-1}(x))$, for all $x \in \mathrm{ind}(g)$, and so $L(\pi g)/_{\equiv}$ is isomorphic to $L(\pi' g)/_{\equiv}$. We thus have that $\pi(g)$ and $\pi'(g)$ are syntactically-equivalent, and the result follows. □

It follows from Lemma 4.14 that the syntactic-equivalence relation of a circuit C constrains the automorphism group of C and the orbits and stabiliser groups of the gates in C. We say that a circuit C is *reduced* if C has trivial syntactic-equivalence classes, i.e. for all gates g and h in C if $g \equiv h$ then $g = h$. An immediate corollary of Lemma 4.14, and an example of this intuitive understanding of the result, is that if a circuit is reduced then it has unique extensions. We now define *transparency* and other properties of circuits in terms of the structure of their syntactic-equivalence classes.

Definition 4.15. Let C be a circuit and g be a gate in C. We say g has *injective labels* if $L(g)$ is an injection. We say g has *unique children* if no two distinct gates in H_g are syntactically-equivalent. We say g has *unique labels* if g has injective labels and unique children.

We say C has *injective labels* (or just C is *injective*) if every gate in C has injective labels. We say C has *unique labels* if every gate in C has unique labels. We say C is *transparent* if every non-trivially invariant gate g in C has unique labels.

It is worth noting that if a circuit has injective labels and is reduced then it has unique labels. The converse is, in general, false.

There is a small technical point that needs some discussion before we continue. The definition of a circuit allows for the inclusion of a gate such that there is no path from this gate to any of the output gates. In this case we can be sure that this gate has no influence on the function computed by the circuit. We say that such a gate is *redundant*.

Definition 4.16. Let C be a circuit and W_t be the transitive closure of the W relation on the gates of C. We say that an internal gate g in C is *redundant* if for all output gates g' in C, we have $\neg W_t(g, g')$.

It is easy to see that we can construct from a given circuit an equivalent circuit without redundant gates by simply removing all of the redundant gates. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a circuit of order n and let g be a redundant gate in C. It is easy to see that g is not the

child of a non-redundant gate, nor is it a relational or output gate. Thus we can define the circuit $C' := \langle G', \Omega, \Sigma\big|_{G'}, \Lambda, L\big|_{G'}\rangle$, where $G' \subseteq G$ is the set of all non-redundant gates in C.

It is easy to see that C' computes the same query as C, and that each $g \in G'$ has unique labels in C if, and only if, g has unique labels in C'. Moreover, it can be shown that if C is symmetric and $g, g' \in G$ then if g and g' are in the same orbit then either both g and g' are in G' or neither are in G'. As such, if C is symmetric then so is C'. We thus have the following Lemma.

Lemma 4.17. *Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a circuit of order n. Let G' be the set of non-redundant gates in C. Let $C' := \langle G', \Omega, \Sigma\big|_{G'}, \Lambda, L\big|_{G'}\rangle$. Then (i) C and C' compute the same query, (ii) no gate in C' is redundant, (iii) if C is symmetric then C' is symmetric, (iv) if C has unique labels then C' has unique labels, (v) if C is reduced then C' is reduced, and (vi) if C is transparent then C' is transparent*

We note that the construction of C' from C in Lemma 4.17 can be implemented by an algorithm that runs in time polynomial in the size of C. As such, for the remainder of this thesis we assume, unless stated otherwise, every circuit has no redundant gates. With this assumption in mind we now show that every circuit with unique children (and so unique labels) has unique extensions.

Proposition 4.18. *Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit of order n. If C has unique children then C has unique extensions.*

Proof. Suppose C has unique children. Let $\pi_\sigma, \pi'_\sigma \in \mathbf{Aut}(C)$ be automorphisms extending $\sigma \in \mathbf{Sym}_n$. We now prove that $\pi_\sigma = \pi'_\sigma$. Let $\pi := \pi'_\sigma \pi_\sigma^{-1}$. We have that π is an automorphism of the circuit and that π extends $\sigma^{-1}\sigma = e$, the identity permutation. It follows that π fixes all input gates and, since for any output gate g, $\pi(g) = \pi(\Omega(\vec{a})) = \Omega(e\vec{a}) = \Omega(\vec{a}) = g$ for all $\vec{a} \in [n]^q$, π fixes all output gates.

We now prove the result by structural induction on the circuit starting from the output gates. Let g be an output gate then $\pi(g) = g$ from the above argument. Let g be an internal, non-output gate. We now show that if $\pi(g) = g$ then for all $h \in H_g$ we have $\pi(h) = h$. Suppose $\pi(g) = g$ and let $h \in H_g$. Since $\pi(g) = g$ we have that $\pi H_g = H_g$, and so $\pi(h) \in H_g$. We have from Lemma 4.14 that h is syntactically-equivalent to $\pi(h)$ and, since g has unique children, h is the only gate in H_g syntactically-equivalent to h. It follows that $\pi(h) = h$.

Let W_t be the transitive closure of the relation W on C. The inductive argument gives us that for all $h \in G$ if there exists an output gate g such that $W_t(h, g)$ or $h = g$, then $\pi(h) = h$. Since the circuit contains no redundant gates, we have that $G = \bigcup_{\vec{a}\in[n]^q}\{g \in G : W_t(g, \Omega(\vec{a}))\}$, and the result follows. □

We have from Proposition 4.18 that circuits with unique gates have unique extensions. However, in order to prove many of the results we will need later we not only need that

each automorphism is uniquely determined by the permutation it extends, but also that the isomorphisms between the labellings of each gate that witness this automorphism are also uniquely determined by the permutation – or, at the very least, well constrained. Without this constraint, computing many properties of the circuit would involve solving some version of the isomorphism problem for τ-structures, where τ is the vocabulary of the gates in question. Indeed, we prove in Chapter 7, that for the class of circuits with unique children deciding if a given function is in fact a valid circuit automorphism is at least as hard as the graph-isomorphism problem. Moreover, we go on to show that a great many important circuit properties (e.g. the orbit of a gate, symmetry of a circuit, etc.) are also at least as hard to decide for circuits with unique gates as the graph isomorphism – and so these properties are, at the very least, not obviously polynomial-time decidable. As such, circuits with unique children, although satisfying the requirement of having unique extensions, are insufficient for our purposes.

We have introduced symmetric circuits and generalised operators and are now ready to state the main result of this thesis.

Theorem 4.19 (Main Theorem)**.** *Let $\mathbf{\Omega}$ be a finite union of* P*-bounded almost relational vectorised operators. Let ρ be a relational vocabulary. Then*

1. *Every query definable in* $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$ *is definable by a* P*-uniform family of transparent symmetric* $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{std}, \rho)$*-circuits, and*
2. *Every query definable by a* P*-uniform family of transparent symmetric* $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{std}, \rho)$*-circuits is definable in* $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$*.*

We note that if the logic $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ is closed under operator quotients then Theorem 4.19 gives us an exact characterisation of $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ in terms of P-uniform families of symmetric circuits. In the remainder of this thesis we build up the necessary tools to prove this result. The formal proof is given in Chapter 9.

4.3 Limitations of Symmetric Bases

In this section we show that any family of symmetric circuits defined over an arbitrary basis of trivially invariant functions can be transformed in polynomial time into a family of symmetric circuits that decide the same query but are defined over the basis $\mathbb{B}_{\mathbf{maj}}$. It follows in particular that it makes no difference to the expressive power of this circuit model whether we consider circuits defined over the basis containing *all* trivially invariant functions or just over $\mathbb{B}_{\mathbf{maj}}$. We may deduce from this observation that in order to construct P-uniform families of symmetric circuits that define queries not in FPC we must allow for bases that include non-trivially invariant functions.

Before we present the novel technical content of this section we should first discuss a related result concerning extensions infinitary logic by families of simple unary Lindström quantifiers established by Kolaitis and Väänänen [34]. We say a Lindström quantifier Q is *simple* and *unary* if the vocabulary of Q consist of a single unary relation and for every pair of appropriate structures $\mathcal{A} = (A, X)$ and $\mathcal{B} = (B, Y)$ if $\mathcal{A} \in \mathcal{G}_Q$, where $\mathcal{G}_Q$ is the class of structures associated with Q, and $|X| = |Y|$ then $\mathcal{B} \in \mathcal{G}_Q$. Kolaitis and Väänänen [34] establish in Proposition 2.7 that if $\mathbb{Q}$ is a family of simple urinary quantifiers then $\mathcal{L}^k(\mathbb{Q}) \leq \mathcal{C}^k$ for all $k < \omega$. We may view Theorem 4.21, the main result of this section, as establishing an analogous result for symmetric circuits, in that it shows that if a query is computable by a family of symmetric circuits over an arbitrary basis of trivially invariant functions then it is computable by a family of symmetric circuits over the standard basis with majority with only a polynomial blowup in size. Importantly, for any fixed basis of trivially invariant functions $\mathbb{B}$ the translation we define that maps symmetric circuits over $\mathbb{B}$ to equivalent symmetric circuits over $\mathbb{B}_{\mathbf{maj}}$ is computable in time polynomial in the size of the input circuit.

Let $F : \{0,1\}^n \to \{0,1\}$ be a trivially invariant function. Recall that the output of F is entirely determined by the number of 1s in its input. Let $c_F \subseteq [n]$ be the set of all $m \leq n$ such that for all $\vec{x} \in \{0,1\}^n$ with m 1s we have $F(\vec{x}) = 1$. Clearly any trivially-invariant function F is entirely determined by c_F. As such, F may be encoded by a tuple $f \in \{0,1\}^n$, where $f(i) = 1$ if, and only if, $i \in c_F$. We assume this encoding below.

Proposition 4.20. *There is a deterministic algorithm that outputs for each trivially invariant function $F : \{0,1\}^n \to \{0,1\}$ (encoded as a binary n-tuple as above) a symmetric circuit C defined over the basis $\mathbb{B}_{\boldsymbol{maj}}$ that computes F. Moreover, this algorithm runs in time polynomial in n and the circuit C has depth at most 5, width at most $2n+2$ and size at most $5n+3$.*

Proof. We have c_F from the input. We now define C. We define the set of gates and wires of C layer by layer as follows. The first layer consists of just the n input gates labelled by the variables $x_1, \ldots, x_n$. The second layer consists of two MAJ gates for each $a \in c_F$, which we denote by $\mathbf{maj}_a$ and $\mathbf{maj}_a^{\neg}$. For each $a \in c_F$ there is one wire from each of $x_1, \ldots, x_n$ to $\mathbf{maj}_a$ and $\mathbf{maj}_a^{\neg}$. For all $a \geq \frac{n}{2}$, there are $2a - n$ wires from 0 to $\mathbf{maj}_a$ and $2a - n + 2$ wires from 0 to $\mathbf{maj}_a^{\neg}$. For all $a < \frac{n}{2}$ there are $n - 2a$ wires from 1 to $\mathbf{maj}_a$ and $n - 2a - 2$ wires from 1 to $maj_a^{\neg}$. The third layer consists of one NOT gate for each $a \in c_F$, which we denote by $\neg_a$. For each $a \in c_F$ there is a wire from $\mathbf{maj}_a^{\neg}$ to $\neg_a$. The fourth layer consists of one AND gate for each $a \in c_F$, which we denote by count_a. For each $a \in c_F$ there is a wire from each of $\mathbf{maj}_a$ and $\neg_a$ to count_a. The fifth layer consists of just a single OR gate, designated as the output gate, and for each $a \in c_F$ there is a wire from count_a to the output gate.

We summarise the circuit up to the fourth layer as follows:

$$\text{count}_a = \begin{cases} \wedge(\mathbf{maj}(x_1,\ldots,x_n,\underbrace{0,\ldots,0}_{2a-n}),\neg(\mathbf{maj}(x_1,\ldots,x_n,\underbrace{0,\ldots,0}_{2a-n+2}))) & a \geq \frac{n}{2} \\ \wedge(\mathbf{maj}(x_1,\ldots,x_n,\underbrace{1,\ldots,1}_{n-2a}),\neg(\mathbf{maj}(x_1,\ldots,x_n,\underbrace{1,\ldots,1}_{n-2a-2}))) & a < \frac{n}{2}. \end{cases}$$

We note that for an input vector $\vec{x}$, $count_a$ evaluates to 1 if, and only if, the number of 1's in $\vec{x}$ equals a. Thus we have that C evaluates to 1 if, and only if, there exists $a \in c_F$ such that the number of 1's in $\vec{x}$ equals a if, and only if, $F(\vec{x}) = 1$. Let $\sigma \in \mathbf{Sym}_n$. We define the automorphism π extending σ as follows. If x_i is an input gate then let $\pi x_i := x_{\sigma i}$. We note that for each majority gate g in the second layer there is exactly one wire from each input gate to g. We note additionally that every other gate in the circuit is connected to the (non-constant) input gates only through a gate in the second layer. As such, if g is an internal gate we let $\pi g := g$, and note that π is an automorphism extending σ. It follows that C is symmetric. It is easy to see that the construction of the circuit C can be implemented by an algorithm running in time polynomial in n.

Furthermore, notice that the first layer contains n gates and the second layer contains at most $2|c_F| \leq 2n$ gates. The third and fourth layer each contain at most n gates. As such C has size at most $n + 2n + 2n + 1 + 2 = 5n + 3$ (the additional 2 is for the constant gates). The width of C is at most $2n + 2$, and the depth is at most 5. □

Theorem 4.21. *Let $\mathbb{B}$ be a basis of trivially invariant functions and let $(C_n)_{n\in\mathbb{N}}$ be a family of symmetric circuits defined over the basis $\mathbb{B}$. Then there exists a family of symmetric circuits $(C'_n)_{n\in\mathbb{N}}$ defined over $\mathbb{B}_{\boldsymbol{maj}}$ that decides the same language. Moreover, the map $C_n \mapsto C'_n$ is polynomial-time computable and for each $n \in \mathbb{N}$, $|C'_n| \leq (5 \cdot |C_n| + 3) \cdot |C_n|$.*

Proof. From C_n we construct C'_n as follows. For each gate $g \in C_n$ labelled by a member of $\mathbb{B}$ we have a symmetric circuit C_g from Proposition 4.20 that computes the same function as g. Then let C'_n be as C_n but with each gate $g \in C_n$ replaced by C_g. It is easy to see that C'_n is symmetric. We note that each gate $g \in C_n$ has $|H_g| \leq |C_n|$, and so the size of C_g is bounded by $5|C_n| + 3$. Thus the size of C'_n is bounded by $(5f(n) + 3)f(n)$. This algorithm clearly runs in polynomial-time. □

This result establishes that for any family of circuits over an arbitrary basis of trivially invariant functions we can construct another family of symmetric circuits over the majority basis computing the same function and with only a polynomial blowup in size. It follows from the fact that this translation is polynomial-time computable that if the first family of symmetric circuits is P-uniform then the second family defined over $\mathbb{B}_{\mathbf{maj}}$ will be as well. As such, we cannot increase the expressive power of the circuit model studied by Anderson and Dawar [3], which is defined in such a way as to necessitate that the circuit be defined

over a basis of trivially invariant functions, by simply considering some alternative basis of trivially invariant functions. This result thus motivates the necessity of the generalisation of the circuit model so as to allow for bases that may include non-trivially invariant functions.

Chapter 5

Translating Formulas to Circuits

There is a standard translation from fixed-point logics to families of symmetric circuits (see [32] for details). However, this translation does not, in general, produce families of *transparent* symmetric circuits. In this chapter we show that for a P-bounded family of almost relational generalised operators $\mathbf{\Omega}$ each formula in $\mathrm{FP}(\widetilde{\mathbf{\Omega}})$ can be translated to a P-uniform family of transparent symmetric circuits defined over the basis $\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}$. This translation suffices to prove one direction in Theorem 4.19.

We build up to this translation as follows. First, in Lemma 5.1 we show that each formula in an extension of first-order logic by a family of many-sorted quantifiers $\mathbf{Q}$ can be translated to a P-uniform family of transparent symmetric circuits over the basis $\mathbb{B}_Q \cup \mathbb{B}_{\mathrm{std}}$. Second, in Proposition 5.3 we use Lemma 5.1 to show that each P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}})$-substitution programs can be translated to a P-uniform family of transparent symmetric circuits over the basis $\mathbb{B}_Q \cup \mathbb{B}_{\mathrm{std}}$. Finally, we use the translation from extensions of fixed-point logic to P-uniform families of substitution programs given in Lemma 3.26, as well as Lemma 5.4, to complete the translation.

Let ρ be a relational vocabulary. Let $\mathbf{Q}$ be a family of quantifiers. We aim to define a translation from P-uniform families of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-substitution programs with constant width to P-uniform families of transparent symmetric $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuits. We say that a $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuit C *translates a* $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$*-formula* $\theta(\vec{x})$ *for* $n \in \mathbb{N}$ if C is a transparent symmetric circuit of order n and for all $\mathcal{A} \in \mathrm{fin}[\rho, n]$, $\gamma \in [n]^{\underline{A}}$, and $\alpha \in A^{\vec{x}}$ we have that $\gamma(\alpha(\vec{x})) \in C[\gamma\mathcal{A}]$ if, and only if, $\mathcal{A} \models \theta[\alpha]$. We say a family of circuits $(C_n)_{n\in\mathbb{N}}$ *translates* $\theta(\vec{x})$ if $(C_n)_{n\in\mathbb{N}}$ is P-uniform and for all $n \in \mathbb{N}$, C_n translates $\theta(\vec{x})$ for n. We now show that for each $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formula $\theta(\vec{x})$ and each $n \in \mathbb{N}$ we can define a circuit C_n that translates $\theta(\vec{x})$ for n. Moreover, we show that the construction of C_n can be implemented by an algorithm and we give precise bounds on the running time of this algorithm.

Lemma 5.1. *Let ρ be a relational vocabulary containing at least one non-nullary symbol. Let $\mathbf{Q}$ be a family of quantifiers. There is a function that takes as input a number $n \in \mathbb{N}$, an $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-formula $\theta(\vec{x})$ and outputs a $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}_{std}, \rho)$-circuit C such that C translates θ for n.*

Moreover, there exists a polynomial p such that this function is computable by an algorithm that for a given input terminates in at most $p(|\theta|n^{\mathrm{width}(\theta)+\mathrm{owidth}(\theta)})$ many steps.

Proof. We assume, without a loss of generality, that at least one relation symbol from ρ appears in θ. If this is not the case we take a conjunction of θ with a tautology of the form $\forall x\,(T(x,\dots,x) \vee \neg T(x,\dots,x)$ for some non-nullary $T \in \rho$. For each $Q \in \mathbf{Q}$ let $\tau_Q := (\boldsymbol{R}_Q, \boldsymbol{S}_Q, \zeta_Q)$ be the vocabulary of Q and let ar_Q be the arity of Q.

There is a natural order on the symbols in the formula θ . We can use the order to define for each many-sorted quantifier Q that appears in θ an injection $f_Q : \boldsymbol{R}_Q \to \mathbb{N}_0$ such that $f_Q(R) \leq |\theta|$ for all $R \in \boldsymbol{R}_Q$.

We structure this proof as follows. First, we define a formula λ such that θ and λ define the same query. We think of λ as a normal form for θ defined so as to remove any "unwanted symmetries" in the structure of the formula that would result in two children of a non-symmetric gate being syntactically equivalent when we translate to circuits. We use λ to define a circuit C and show that C is symmetric, transparent, and defines the same query as λ (and hence θ) for structures of size n. Lastly, we show that we can construct C within the required time bounds.

Before we define λ we first define a few auxiliary formulas. Let $T \in \rho$ be a non-nullary symbol. For a variable y let $\textsc{no-op}_y \equiv (T(y\ \dots, y) \vee (\neg T(y,\dots,y)))$ and let $\textsc{no-op-all} \equiv \forall u\, \textsc{no-op}_u$. Let $\vec{y} := (y_1,\dots,y_m)$ be a (possibly empty) sequence of variables. If $\vec{y}$ is not empty let $\textsc{tag}_{\vec{y}} \equiv (\textsc{no-op}_{y_1} \wedge (\textsc{no-op}_{y_2} \wedge (\dots \wedge (\textsc{no-op}_{y_m})\dots))\dots))$. If $\vec{y}$ is empty let $\textsc{tag}_{\vec{y}} \equiv \forall u\,((u = u) \wedge (u = u))$. We define a similar helper formula $\textsc{tag-num}$ for each $e \in \mathbb{N}$

$$\textsc{tag-num}_e \equiv \underbrace{(\textsc{no-op-all} \wedge (\textsc{no-op-all} \wedge (\dots \wedge (\textsc{no-op-all})\dots)))}_{e \text{ times}}.$$

It is easy to see that $\textsc{tag}_{\vec{y}}$ and $\textsc{tag-num}_e$ are tautologies for any $e \in \mathbb{N}$ and sequence of variables $\vec{y}$. Notice that for $e, d, \in \mathbb{N}$, $\textsc{tag-num}_e = \textsc{tag-num}_d$ if, and only if, $e = d$. Moreover, for sequences of variables $\vec{y}$ and $\vec{z}$, $\textsc{tag}_{\vec{y}} = \textsc{tag}_{\vec{z}}$ if, and only if, $\vec{z} = \vec{y}$.

Let ψ be a subformula of $\theta(\vec{x})$ of the form $Q[(\vec{y}_1^R,\dots,\vec{y}_{r_R}^R)\,\psi_R]_{R\in\boldsymbol{R}_Q}$. For each $R \in \boldsymbol{R}_Q$ let

$$\psi'_R \equiv ((\forall u\,(u = u)) \wedge \psi_R) \wedge (\textsc{tag-num}_{f_Q(R)} \wedge \textsc{tag}_{\vec{y}_1^R \dots \vec{y}_{r_R}^R})$$

and let

$$\psi'(\vec{y}) \equiv Q[(\vec{y}_1^R,\dots,\vec{y}_{r_R}^R)\,\psi'_R]_{R\in\boldsymbol{R}_Q}.$$

It follows from the fact that ψ' is defined from ψ by taking conjunctions with tautologies that ψ and ψ' define the same query. Let θ' be defined from θ by recursively replacing each subformula ψ in θ with a many-sorted quantifier at its head with the corresponding formula ψ'. Let $\lambda \equiv \theta' \wedge \textsc{tag}_{\vec{x}}$. It can be shown by induction that $\lambda(\vec{x})$ and $\theta(\vec{x})$ define the same

query. It can also be shown by induction that $\text{width}(\lambda) \leq 1 + \text{owidth}(\theta) + \text{width}(\theta)$ and that $\text{owidth}(\lambda) \leq \text{owidth}(\theta) + 1$.

We fix $n \in \mathbb{N}$. We now define a binary relation on pairs of $\text{FO}(\mathbf{Q})[\rho]$-formulas and variable assignments. Let $\phi_1, \phi_2 \in \text{FO}(\mathbf{Q})[\rho]$. Let $\vec{x}_1$ and $\vec{x}_2$ be sequences of variables such that $\text{free}(\phi_1) \subseteq \vec{x}_1$ and $\text{free}(\phi_2) \subseteq \vec{x}_2$ and let $\alpha_1 \in [n]^{\vec{x}_1}$ and $\alpha_2 \in [n]^{\vec{x}_2}$. Let $c = \{c_0, \ldots, c_n\}$ be a set of constant symbols and for each $z \in [2]$ let $\alpha_z^c : \vec{x}_z \to c$ be defined such that for all $x \in \vec{x}_z$, $\alpha_z^c(x) = c_{\alpha_z(x)}$. Let $(\phi_1, \alpha_1) \sim (\phi_2, \alpha_2)$ if, and only if,

- ϕ_1 is of the form $x_1 = y_1$ and ϕ_2 is of the form $x_2 = y_2$ and $\alpha_1(x_1) = \alpha_1(y_1)$ if, and only if, $\alpha_2(x_2) = \alpha_2(y_2)$, or
- the formulas $\phi_1[\frac{\alpha_1^c(\vec{x}_1)}{\vec{x}_1}]$ and $\phi_2[\frac{\alpha_2^c(\vec{x}_2)}{\vec{x}_2}]$ are equal up to renaming of bound variables.

It can be seen that $\sim$ is an equivalence relation. For a formula $\phi \in \text{FO}(\mathbf{Q})[\rho]$ and an assignment $\alpha \in [n]^{\vec{z}}$ where $\text{free}(\phi) \subseteq \vec{z}$ let $[\phi, \alpha]$ be the equivalence class of (ϕ, α).

Example 5.2. Let T be a ternary relation symbol and let $\phi_1(z, w) \equiv \exists u\, T(z, w, u)$. Let $\phi_2(x) \equiv \exists v\, T(x, x, v)$ and let α_1 and α_2 be assignments such that $\alpha_1(z) = \alpha_1(w) = 1$ and $\alpha_2(x) = 1$. Then $\phi_1[\frac{\alpha_1^c(z)}{z}\frac{\alpha_1^c(w)}{w}] \equiv \exists u\, T(c_1, c_1, u)$ and $\phi_2[\frac{\alpha_2^c(x)}{x}] \equiv \exists v\, T(c_1, c_1, v)$. These two formulas are equal up to renaming of bound variables and so $(\phi_1, \alpha_1) \sim (\phi_2, \alpha_2)$. Notice that if $\phi_1(x, y)$ were instead defined such that $\phi_1(x, y) \equiv \exists u\, T(u, z, w)$ and ϕ_2, α_1, and α_2 were defined as above, then $(\phi_1, \alpha_1) \not\sim (\phi_2, \alpha_2)$.

Notice that for $a, b \in \mathbb{N}$ and any assignments α and β we have $(\text{TAG-NUM}_a, \alpha) \sim (\text{TAG-NUM}_b, \beta)$ if, and only if, $a = b$. For sequences of variables $\vec{x}$ and $\vec{y}$ and assignments $\alpha \in [n]^{\vec{x}'}$ and $\beta \in [n]^{\vec{y}'}$ where $\vec{x}'$ and $\vec{y}'$ are sequences of variables such that $\vec{x} \subseteq \vec{x}'$ and $\vec{y} \subseteq \vec{y}'$ it follows that $(\text{TAG}_{\vec{x}}, \alpha) \sim (\text{TAG}_{\vec{y}}, \beta)$ if, and only if, $\alpha(\vec{x}) = \beta(\vec{y})$.

For each $\psi \in \text{cl}(\lambda)$ let $G_\psi := \{g_{[\psi,\alpha]} : \alpha \in [n]^{\text{free}(\psi)}\}$. Let $G := \bigcup_{\psi \in \text{cl}(\lambda)} G_\psi$. We define Σ, Λ and L for $g := g_{[\psi,\alpha]} \in G$ as follows.

- If ψ is of the form $x = y$ let $\Sigma(g) = 1$ if, and only if, $\alpha(x) = \alpha(y)$.
- If ψ is of the form $T(\vec{y})$ for some $T \in \rho$ let $\Sigma(g) = T$ and let $\Lambda_T(g) = \alpha(\vec{y})$.
- Suppose ψ is of the form $Q[(\vec{y}_1^R, \ldots, \vec{y}_{r_R}^R) \cdot \psi_R]_{R \in \boldsymbol{R}_Q}$ for some $Q \in \mathbf{Q}$. Let $\mathcal{G}$ be the class of τ_Q-structures associated with Q. Let $\Sigma(g) = F_{\mathcal{G}}[\text{ar}_Q]$. Let $L(g) : \tau_Q[\text{ar}_Q] \to G$ be defined as follows. Let $(\vec{a}, R) \in \tau_Q[\text{ar}_Q]$. For each $i \in [r_R]$ there exists $\vec{a}_i \in [n]^{\text{ar}(\zeta_Q(R)(i))}$ such that $\vec{a} = (\vec{a}_1, \ldots, \vec{a}_{r_R})$. Let $L(g)(\vec{a}, R) = g_{[\psi_R,\beta]}$ where $\beta = \alpha\frac{\vec{a}_1}{\vec{y}_1^R} \ldots \frac{\vec{a}_{r_R}}{\vec{y}_{r_R}^R}$.
- Suppose ψ is of the form $Qz\, \phi(\vec{y}, z)$ for $Q \in \{\forall, \exists\}$, then if $Q = \forall$ let $\Sigma(g) = \text{AND}[n]$ and otherwise let $\Sigma(g) = \text{OR}[n]$. Let $L(g) : [n] \to G$ be defined for $i \in [n]$ by $L(g)(i) = g_{[\phi,\beta]}$, where $\beta := \alpha\frac{i}{z}$.
- If ψ is of the form $\phi_1 \wedge \phi_2$ let $\Sigma(g) = \text{AND}[2]$ and let $L(g) : [2] \to G$ be defined for $i \in [2]$ by $L(g)(i) = g_{[\phi_i,\beta_i]}$ where $\beta_i = \alpha|_{\text{free}(\phi)_i}$. The same approach is used for the disjunctive case.

- If ψ is of the form $\neg\phi$ let $\Sigma(g) = \text{NOT}$ and $L(g) : [1] \to G$ be defined by $L(g)(1) = g_{[\phi,\alpha]}$.

For a given $g \in G$ by checking cases we can show that $L(g)$ is an injection. Let $q = |\vec{x}|$, where $\vec{x}$ is the sequence of variables free in λ. Let $\Omega : [n]^q \to G$ be defined for $\vec{a} \in [n]^q$ by $\Omega(\vec{a}) = g_{[\lambda,\frac{\vec{a}}{\vec{x}}]}$. It follows from the definition of λ that for all $\vec{a} \in [n]^q$ we have that $g_{[\lambda,\frac{\vec{a}}{\vec{x}}]}$ is not a constant gate. Let $\vec{a}, \vec{b} \in [n]^q$ and suppose $\Omega(\vec{a}) = \Omega(\vec{b})$. Then $g_{[\lambda,\frac{\vec{a}}{\vec{x}}]} = g_{[\lambda,\frac{\vec{b}}{\vec{x}}]}$ and so $(\lambda, \frac{\vec{a}}{\vec{x}}) \sim (\lambda, \frac{\vec{b}}{\vec{x}})$. It follows from the structure of λ that $\vec{a} = \vec{b}$. We conclude that Ω is injective. Let $T \in \rho$ and let $g_{[\psi,\alpha]}, g_{[\phi,\beta]} \in G$ be such that $\Lambda_T(g_{[\psi,\alpha]}) = \Lambda_T(g_{[\phi,\beta]})$. Then ψ is of the form $T(\vec{z})$ and ϕ is of the form $T(\vec{w})$ and $\alpha(\vec{z}) = \Lambda_T(g_{[\psi,\alpha]}) = \Lambda_T(g_{[\phi,\beta]}) = \beta(\vec{w})$. It follows that $(\psi, \alpha) \sim (\phi, \beta)$ and so $g_{[\psi,\alpha]} = g_{[\phi,\beta]}$. So we have that Λ_T is injective. We conclude that $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ is a circuit of order n with injective labels.

Claim 5.2.1. *The circuit C is symmetric.*

Proof. Let $\sigma \in \mathbf{Sym}_n$. Let $\pi_\sigma : G \to G$ be defined such that $\pi_\sigma g_{[\psi,\alpha]} = g_{[\psi,\sigma\alpha]}$ for each $g_{[\psi,\alpha]} \in G$. It can be shown that π_σ is a bijection. We now show that π_σ is an automorphism of C extending σ. We prove this by induction on the structure of the circuit. Let $g_{[\psi,\alpha]} \in G$.

Suppose $g_{[\psi,\alpha]}$ is an input gate. If $g_{[\psi,\alpha]}$ is a constant gate then ψ is of the form $y_1 = y_2$ and since σ is a bijection we have $\alpha(y_1) = \alpha(y_2)$ if, and only if, $\sigma\alpha(y_1) = \sigma\alpha(y_2)$. It follows that $(\psi, \alpha) \sim (\psi, \sigma\alpha)$ and so $\pi_\sigma g_{[\psi,\alpha]} = g_{[\psi,\sigma\alpha]} = g_{[\psi,\alpha]}$. If $g_{[\psi,\alpha]}$ is a relational gate then ψ is of the form $T(\vec{y})$ for some relation symbol T and $\Lambda_T(g_{[\psi,\alpha]}) = \alpha(\vec{y})$. It follows that $\Lambda_T(\pi_\sigma g_{[\psi,\alpha]}) = \Lambda_T(g_{[\psi,\sigma\alpha]}) = \sigma\alpha(\vec{y}) = \sigma\Lambda_T(g_{[\psi,\alpha]})$.

Suppose $g_{[\psi,\alpha]}$ is an internal gate. It can be shown by considering each of the cases in the definition of the circuit that for each $g_{[\phi,\alpha]} \in H_{g_{[\psi,\alpha]}}$ we have $\pi_\sigma g_{[\phi,\alpha]} = g_{[\phi,\sigma\alpha]} \in H_{g_{[\psi,\sigma\alpha]}} = H_{\pi_\sigma g_{[\psi,\alpha]}}$. From this we have $\pi_\sigma H_{g_{[\psi,\alpha]}} = H_{\pi_\sigma g_{[\psi,\alpha]}}$. If $g_{[\psi,\alpha]}$ is a symmetric gate, then this is sufficient to conclude that $\pi_\sigma L(g_{[\psi,\alpha]})$ is isomorphic to $L(\pi_\sigma g_{[\psi,\alpha]})$. Suppose $g_{[\psi,\alpha]}$ is a non-symmetric gate. Then ψ is of the form $Q[(\vec{y}_1^R, \ldots, \vec{y}_{r_R}^R)\, \psi_R]_{R \in \boldsymbol{R}_Q}$ for some $Q \in \mathbf{Q}$. Let $\mathcal{G}$ be the class of structures associated with Q. For each $(\vec{a}, R) \in \tau_Q[\text{ar}_Q]$ and each $i \in [r_R]$ let $\vec{a}_i \in [n]^{\text{ar}(\zeta_Q(R)(i))}$ be such that $\vec{a} = (\vec{a}_1 \ldots \vec{a}_{r_R})$ and let

$$\beta_{\alpha,\vec{a}} := \alpha \frac{\vec{a}_1}{\vec{y}_1^R} \ldots \frac{\vec{a}_{r_R}}{\vec{y}_{r_R}^R} \tag{5.1}$$

and let $\sigma\vec{a} = (\sigma\vec{a}_1, \ldots, \sigma\vec{a}_{r_R})$. Let $(\vec{a}, R) \in \tau_Q[\text{ar}_Q]$. Notice that

$$\sigma\beta_{\alpha,\vec{a}} = \sigma(\alpha \frac{\vec{a}_1}{\vec{y}_1^R} \ldots \frac{\vec{a}_{r_R}}{\vec{y}_{r_R}^R}) = (\sigma\alpha)\frac{\sigma\vec{a}_1}{\vec{y}_1^R} \ldots \frac{\sigma\vec{a}_{r_R}}{\vec{y}_{r_R}^R} = \beta_{\sigma\alpha,\sigma\vec{a}}.$$

Then $\pi_\sigma L(g_{\psi,\alpha})(\vec{a}, R) = \pi_\sigma g_{[\phi,\beta_{\alpha,\vec{a}}]} = g_{[\phi,\sigma\beta_{\alpha,\vec{a}}]} = g_{[\phi,\beta_{\sigma\alpha,\sigma\vec{a}}]} = L(g_{\psi,\sigma\alpha})(\sigma\vec{a}, R)$. It follows that $\pi_\sigma L(g_{[\psi,\alpha]})$ is isomorphic to $L(\pi_\sigma g_{[\psi,\alpha]})$.

Suppose $g_{[\psi,\alpha]}$ is an output gate. Then $\psi \equiv \lambda(\vec{x})$. It follows that $\pi_\sigma\Omega(\alpha(\vec{x})) = \pi_\sigma g_{[\psi,\alpha]} = g_{[\psi,\sigma\alpha]} = \Omega(\sigma\alpha(\vec{x}))$. This completes the proof of the Claim 5.2.1.

□

Claim 5.2.2. *Let $\mathcal{A} \in \text{fin}[\rho, n]$ be a structure and let $\gamma \in [n]^{\underline{A}}$. For each $g_{[\psi,\alpha]} \in G$ we have $\mathcal{A} \models \psi[\gamma^{-1}\alpha]$ if, and only if, $C[\gamma\mathcal{A}](g_{[\psi,\alpha]}) = 1$.*

Proof. We prove this claim by induction on the structure of the circuit. Let $g_{[\psi,\alpha]} \in G$. If $g_{[\psi,\alpha]}$ is an input gate then ψ has at its head a relation symbol or equality. It is easy to prove the claim in either case. Suppose $g_{[\psi,\alpha]}$ is an internal gate and the claim holds for each child of $g_{[\psi,\alpha]}$. If $g_{[\psi,\alpha]}$ is labelled by an element of $\mathbb{B}_{\text{std}}$ then ψ has at its head either a Boolean connective, an existential, or a universal quantifier. It can be shown in each of these cases that $\mathcal{A} \models \psi[\gamma^{-1}\alpha]$ if, and only if, $C[\gamma\mathcal{A}](g_{[\psi,\alpha]}) = 1$.

Suppose $g_{[\psi,\alpha]}$ is labelled by an element of $\mathbb{B}$. Then ψ is of the form $Q[(\vec{y}_1^R, \ldots, \vec{y}_{r_R}^R)\psi_R]_{R \in \boldsymbol{R}_Q}$ for some $Q \in \mathbf{Q}$. Let $\mathcal{G}$ be the set of structures associated with Q and let $\mathcal{I}$ be the $L[\rho, \tau_Q]$-interpretation defined by ψ. Let $\mathcal{B} = \mathcal{I}(\mathcal{A}, \gamma^{-1}\alpha)$. Let $(\vec{a}, R) \in \tau_Q[\text{ar}_Q]$. Let $\beta_{\alpha,\vec{a}}$ be defined as in Equation 5.1. From the induction hypothesis $L^{\gamma\mathcal{A}}(g_{[\psi,\alpha]})(\vec{a}, R) = C[\gamma\mathcal{A}](g_{[\psi_R,\beta_{\alpha,\vec{a}}]}) = 1$ if, and only if, $\mathcal{A} \models \psi_R[\gamma^{-1}\beta_{\alpha,\vec{a}}]$ if, and only if, $\gamma^{-1}\vec{a} \in R^{\mathcal{B}}$. It follows that $\mathcal{B}$ and $L^{\gamma\mathcal{A}}(g_{[\psi,\alpha]})$ are isomorphic structures and so $C[\gamma\mathcal{A}](g_{[\psi,\alpha]}) = 1$ if, and only if, $F_{\mathcal{G}}[\text{ar}_Q](L^{\gamma\mathcal{A}}(g_{[\psi,\alpha]})) = F_{\mathcal{G}}[\text{ar}_Q](\mathcal{B}) = 1$ if, and only if, $\mathcal{B} \in \mathcal{G}$ if, and only if, $\mathcal{A} \models \psi[\gamma^{-1}\alpha]$. Claim 5.2.2 follows by induction.

□

Let $\mathcal{A} \in \text{fin}[\rho, n]$, let $\gamma \in [n]^{\underline{A}}$ and let $\alpha \in A^{\vec{x}}$. Then $g_{[\lambda,\gamma\alpha]} = \Omega(\gamma\alpha(\vec{x}))$. It follows from Claim 5.2.2 that $C[\gamma\mathcal{A}](\Omega(\gamma\alpha(\vec{x})) = 1$ if, and only if, $\mathcal{A} \models \lambda[\alpha]$. In other words, C and $\theta(\vec{x})$ express the same query for structures of size n.

Claim 5.2.3. *The circuit C is transparent.*

Proof. If every gate in C is symmetric then C is transparent. Suppose there exists a non-symmetric gate $g_{[\psi,\alpha]} \in G$. Then ψ is of the form $Q[(\vec{y}_1^R, \ldots, \vec{y}_{r_R}^R)\psi_R]_{R \in \boldsymbol{R}_Q}$ for some $Q \in \mathbf{Q}$. Let $(\vec{a}^1, R_1), (\vec{a}^2, R_2) \in \tau_Q[\text{ar}_Q]$. Let r_1 be the arity of R_1 and r_2 be the arity of R_2. For each $i \in [2]$ and $j \in [r_i]$ let $\vec{y}_j^i := \vec{y}_j^{R_i}$. For each $i \in [2]$ and each $j \in [r_R]$ there exists $\vec{a}_i^j \in [n]^{\text{ar}(\zeta_Q(R_i)(j))}$ such that $\vec{a}^i = (\vec{a}_1^i \ldots \vec{a}_{r_i}^i)$. For each $i \in [2]$ let $\beta_i = \alpha\frac{\vec{a}_1^i}{\vec{y}_1^i} \ldots \frac{\vec{a}_{r_i}^i}{\vec{y}_{r_i}^i}$ and let $h_i := L(g_{[\psi,\alpha]})(\vec{a}^i, R_i) = g_{[\psi_{R_i},\beta_i]}$. Suppose $h_1 \equiv h_2$. We aim to show that $h_1 = h_2$.

From the definition of λ it follows that $\psi_{R_i} \equiv \kappa_i^1 \wedge \kappa_i^2$ where $\kappa_i^1 \equiv ((\forall u.u = u) \wedge \psi_i$ and $\kappa_i^2 \equiv (\text{TAG}_{\vec{y}_1^i \ldots \vec{y}_{r_i}^i} \wedge \text{TAG-NUM}_{f_Q(R_i)}$ for some formula ψ_i. For each $i \in [2]$ it follows from the construction of the circuit that $g_{[\kappa_i^1,\beta_i]}$ has a grandchild that is a constant gate while no grandchild of $g_{[\kappa_i^2,\beta_i]}$ is a constant gate. As such, for all $i, j \in [2]$ we have $g_{[\kappa_i^1,\beta_i]} \not\equiv g_{[\kappa_j^2,\beta_j]}$. It follows from the fact that $h_1 \equiv h_2$ that $g_{[\kappa_1^1,\beta_1]} \equiv g_{[\kappa_2^1,\beta_2]}$ and $g_{[\kappa_1^2,\beta_1]} \equiv g_{[\kappa_2^2,\beta_2]}$.

Let $i \in [2]$. We have $\kappa_i^2 \equiv \epsilon_i^1 \wedge \epsilon_i^2$ where $\epsilon_i^1 \equiv \text{TAG}_{\vec{y}_1^i \ldots \vec{y}_{r_i}^i}$ and $\epsilon_i^2 \equiv \text{TAG-NUM}_{f_Q(R_i)}$. Let $j \in [2]$. Then $g_{[\epsilon_i^1,\beta_i]}$ is an OR-gate and $g_{[\epsilon_j^2,\beta_j]}$ is an AND-gate, and so $g_{[\epsilon_i^1,\beta_i]} \not\equiv g_{[\epsilon_j^2,\beta_j]}$. It follows from the fact that $g_{[\kappa_1^2,\beta_1]} \equiv g_{[\kappa_2^2,\beta_2]}$ that $g_{[\epsilon_1^1,\beta_1]} \equiv g_{[\epsilon_2^1,\beta_2]}$ and $g_{[\epsilon_1^2,\beta_1]} \equiv g_{[\epsilon_2^2,\beta_2]}$.

Notice that for each $e \in \mathbb{N}$, TAG-NUM$_e$ defines an e-length sequence of nested tautologies. This is translated to a circuit consisting of an e-height tower of tautologies. As such, it can be shown that since $g_{[\epsilon_1^2,\beta_1]} \equiv g_{[\epsilon_2^2,\beta_2]}$ we have $f_Q(R_1) = f_Q(R_2)$. Moreover, since f_Q is an injection and so $R_1 = R_2$. Similarly, it follows from the definition of TAG and the fact that $g_{[\epsilon_1^1,\beta_1]} \equiv g_{[\epsilon_2^1,\beta_2]}$ that $\beta_1 = \beta_2$ and so $\vec{a}^1 = \vec{a}^2$. We thus conclude that $h_1 = L(g_{[\psi,\alpha]})(\vec{a}^1, R_1) = L(g_{[\psi,\alpha]})(\vec{a}^2, R_2) = h_2$.

It follows that no two children of $g_{[\psi,\alpha]}$ are syntactically-equivalent and since $L(g_{[\psi,\alpha]})$ is injective it follows that $g_{[\psi,\alpha]}$ has unique labels. We conclude that C is transparent. This completes the proof of Claim 5.2.3. □

It can be shown that $|\lambda| \leq c_1|\theta|n^{\text{owidth}(\theta)+\text{width}(\theta)+1}$ for a constant c_1. We can construct the set of gates by iterating over the subformulas of λ and for each subformula ψ and assignment $\alpha \in [n]^{\text{free}(\psi)}$ defining a gate $g_{[\psi,\alpha]}$. We can thus construct the set of gates in time polynomial in $|\lambda|n^{\text{width}(\lambda)}$ and we note that $|\lambda|n^{\text{width}(\lambda)} \leq c_1|\theta|n^{2(\text{owidth}(\theta)+\text{width}(\theta)+1)}$. Since the rest of the circuit can be constructed in time polynomial in the number of gates, it follows that the construction of C can be completed in time polynomial in $|\theta|n^{\text{owidth}(\theta)+\text{width}(\theta)}$. We have from the above three claims that C translates λ (and hence θ) for n, and from the above argument we can construct C within the required time bounds. This completes the proof of the lemma. □

Let ρ be a relational vocabulary. Let $(\Phi_n)_{n\in\mathbb{N}}$ be a P-uniform family of FO($\mathbf{Q}$)[ρ]-substitution programs. Let $n \in \mathbb{N}$ and let $(\phi_1, \ldots, \phi_k) = \Phi_n$. For each $i \in [k]$ we can treat the second-order variables in ϕ_i as relation symbols and use Lemma 5.1 to define a circuit $C_{n,i}$ that translates ϕ_i for n. We can connect these circuits in order to form a single circuit C_n that translates the flattening of Φ_n for n. In this way we can define a P-uniform family of transparent symmetric circuits $(C_n)_{n\in\mathbb{N}}$ that decides the same query as $(\Phi_n)_{n\in\mathbb{N}}$. We now formalise this argument and prove the following proposition.

Proposition 5.3. *Let ρ be a relational vocabulary. Let $\mathbf{Q}$ be a set of many-sorted quantifiers. Each query definable by a* P*-uniform family of* FO($\mathbf{Q}$)[ρ]*-substitution programs with constant width is definable by a* P*-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}_{std}, \rho)$-circuits.*

Proof. Let $\Theta := (\Theta_n)_{n\in\mathbb{N}}$ be a P-uniform family of FO($\mathbf{Q}$)[ρ] substitution programs. Let $n \in \mathbb{N}$. Let $k = |\Theta_n|$ and let $(\theta_{n,1}, \ldots, \theta_{n,k}) := \Theta_n$. For each $i \in [k]$ let $\vec{V_i}$ be the element-sort second-order variables that appear in $\theta_{n,i}$. We treat these second-order variables as relation symbols, and suppose, without a loss of generality, that for all $i \in [k]$ none of the variables in $\vec{V_i}$ are in ρ. For each $i \in [k]$ let $\rho_i = \vec{V_i} \cup \rho$.

Suppose ρ is empty. It follows that all ρ-structures of the same size are isomorphic. We can evaluate Θ_n on the ρ-structure with universe $[n]$ and construct a transparent symmetric circuit C_n that decides the same query as Θ_n for structures of size n.

Suppose all of the relation symbols in ρ are nullary. Then each ρ-structure of size n is in one of $2^{|\rho|}$ many isomorphism classes. We can evaluate Θ_n on a structure of size n from each of these classes and construct a single transparent symmetric circuit C_n that decides the same query as Θ_n for structures of size n.

We suppose that at least one relation symbol in ρ is not nullary. From Lemma 5.1 we may construct for each $i \in [k]$ a $(\mathbb{B}, \rho_i)$-circuit $C_{n,i}$ such that $C_{n,i}$ translates the formula $\theta_{n,i}$ for n. We assume, without a loss of generality, that in each of these circuits none of the input gates are also output gates (if this is not the case, we can alter the circuit by adding in single-input AND-gates with each gate that is both an input and output gate taken as input to the AND-gate, and the AND-gate then assigned to be an output gate).

For each $i \in [k]$ let $\theta'_{n,i}$ be the flattening of Θ_n at i. We recall that for each $i \in [k]$ the formula $\theta'_{n,i}$ is defined by replacing each appearance of a symbol V_j in $\theta_{n,i}$ with the formula $\theta'_{n,j}$. Notice, this definition is a (backwards) recursive definition. We will similarly define $C'_{n,i}$ from $C_{n,i}$ by replacing each input gate labelled by the symbol V_j and tuple $\vec{a}$ with the output gate of $C'_{n,j}$ labelled by $\vec{a}$. For each $i \in [k]$ let $(G_i, \Omega_i, \Sigma_i, \Lambda_i, L_i) := C_{n,i}$. We now define the $(\mathbb{B}, \rho)$-circuit $C'_{n,i} := (G'_i, \Omega'_i, \Sigma'_i, \Lambda'_i, L'_i)$ recursively. If $i = k$ then $C'_{n,i} = C_{n,i}$. If $i < k$ then we define $C'_{n,i}$ from the circuits $\{C'_{n,j} : j > i\}$ as follows.

- Let $G'_i = \{g \in G_i : \Sigma(g) \notin \vec{V_i}\} \cup \bigcup_{V_j \in \vec{V}} G'_j$. We identify input gates labelled by the same relation symbol and tuple so that there are no duplicate input gates.
- Let Λ_i be defined for each $T \in \rho$, $\vec{a} \in [n]^{\mathrm{arty}(T)}$, and $g \in G'_i$ such that $(\Lambda'_i)_T(g) = \vec{a}$ if, and only if, (i) $g \in G_i$ and $(\Lambda_i)_T(g) = \vec{a}$ or (ii) there exists $j > i$ such that $g \in G'_j$ and $(\Lambda'_j)_T(g) = \vec{a}$.
- For each $g \in G'_i$ if $g \in G_i$ let $\Sigma'_i(g) = \Sigma_i(g)$ and otherwise there exists $j > i$ such that $g \in G'_j$ and let $\Sigma'_i(g) = \Sigma'_j(g)$.
- Let q be the arity of the query decided by $C_{n,i}$. Let Ω'_i be an injection from $[n]^q$ to G'_i defined such that $\Omega'_i(\vec{a}) = \Omega_i(\vec{a})$ for all $\vec{a} \in [n]^q$.
- Let $g \in G'_i$. Suppose $g \in G_i$. Let $L'_i(g) : \mathbf{Dom}(L_i(g)) \to G'_i$ be defined for $a \in \mathbf{Dom}(L_i(g))$ such that $L'_i(g)(a) = L_i(g)(a)$ if $\Sigma_i(L_i(g)(a)) \notin \vec{V_i}$ and otherwise let $L'_i(g)(a) = \Omega_j((\Lambda_i)_{V_j}(L_i(g)(a)))$, where $j > i$ and $\Sigma_i(L_i(g)(a)) = V_j$. Suppose $g \notin G_i$. Then there exists $j > i$ such that $g \in G'_j$. Let $L_i(g) = L'_j(g)$.

We next show, by backwards induction, that for all $i \in [k]$, $C'_{n,i}$ translates $\theta'_{n,i}$ for n. Since $C'_{n,k} = C_{n,k}$ and $\theta'_{n,k} = \theta_{n,k}$, clearly $C'_{n,k}$ translates $\theta'_{n,k}$ for n. Let $i \in [k]$ and suppose for each $j > i$ we have that $C'_{n,j}$ translates $\theta'_{n,j}$ for n. We now show that $C'_{n,i}$ translates $\theta'_{n,i}$ for n. We split the proof over the following three claims.

Claim 5.3.1. *The circuit $C'_{n,i}$ is symmetric.*

Proof. Let $\sigma \in \mathbf{Sym}_n$. Since $C_{n,i}$ is symmetric there exists an automorphism π_i of $C_{n,i}$ extending σ. From the induction hypothesis we have for each $j > i$ that there exists an

automorphism π'_j of $C'_{n,j}$ extending σ. Let $\pi'_i : G'_i \to G'_i$ be defined for each $g \in G'_i$ such that if $g \in G_i$ then $\pi'_i(g) := \pi_i(g)$ and otherwise $\pi'_i(g) := \pi'_j(g)$, where $j > i$ is such that $g \in G'_j$. We now show that π'_i is an automorphism of $C'_{n,i}$ extending σ.

We show for each internal gate g we have that $\pi'_i L'_i(g)$ is isomorphic to $L'_i(\pi'_i g)$. Let g be an internal gate in C'_i. Suppose $g \notin G_i$. Then $g \in G'_j$ for some $j > i$. Since $C'_{n,j}$ is symmetric there exists $\lambda \in \mathbf{Aut}(g)$ such that $\pi'_j L'_j(g) = L_j(\pi'_j g)\lambda$. Then for all $a \in \mathrm{ind}(g)$ we have $\pi'_i L'_i(g)(a) = \pi'_j L'_j(g)(a) = L'_j(\pi'_j g)(\lambda a) = L'_i(\pi'_i g)(\lambda a)$. It follows that $\pi'_i L'_i(g)$ is isomorphic to $L'_i(\pi'_i g)$. Suppose $g \notin G'_j$ for any $j > i$. Then $g \in G_i$. Since $C_{n,i}$ is symmetric there exists $\lambda \in \mathbf{Aut}(g)$ such that $\pi_i L_i(g) = L_i(\pi_i g)\lambda$. Let $a \in \mathrm{ind}(g)$. If $L'_i(g)(a) \in G_i$ then $L_i(g)(a) = L'_i(g)(a)$ and so $\pi'_i L'_i(g)(a) = \pi_i L_i(g)(a) = L_i(\pi_i g)(\lambda a) = L'_i(\pi'_i g)(\lambda a)$. If $L'_i(g)(a) \notin G_i$ then $L'_i(g)(a) = \Omega'_j((\Lambda_i)_{V_j}(L_i(g)(a)))$ for some $j > i$ and $\pi'_i L'_i(g)(a) = \pi'_j \Omega'_j((\Lambda_i)_{V_j}(L_i(g)(a))) = \Omega'_j(\sigma(\Lambda_i)_{V_j}(L_i(g)(a))) = \Omega'_j((\Lambda_i)_{V_j}(\pi_i L_i(g)(a)) = \Omega'_j(((\Lambda_i)_{V_j}(L_i(\pi_i g)(\lambda a))) = L'_i(\pi_i g)(\lambda a) = L'_i(\pi'_i g)(\lambda a)$. It is easy to check the remaining requirements for π'_i to be an automorphism of $C'_{n,i}$ extending σ. This completes the proof of Claim 5.3.1. □

Claim 5.3.2. *The circuit $C'_{n,i}$ is transparent.*

Proof. If all of the gates in $C'_{n,i}$ are symmetric then $C'_{n,i}$ is transparent. Suppose there exists a non-symmetric gate $g \in G'_i$. If $g \in G'_j$ for some $j > i$ then, since $C'_{n,j}$ is transparent it follows that g has unique labels. Otherwise $g \in G_i$. It follows from the construction of $C_{n,i}$ in Lemma 5.1 and from an argument similar to the one used to prove Claim 5.2.3 that g has unique labels. It follows that $C'_{n,i}$ is transparent. □

Claim 5.3.3. *Let $\mathcal{A} \in \mathrm{fin}[\rho, n]$, let $\vec{x}$ be the free variables in $\theta'_{n,i}$, and let $\alpha \in A^{\vec{x}}$. Let $\gamma \in [n]^{\underline{A}}$. Then $C'_{n,i}[\gamma\mathcal{A}](\Omega'_i(\gamma(\alpha(\vec{x})))) = 1$ if, and only if, $\mathcal{A} \models \theta'_{n,i}[\alpha]$.*

Proof. Let $\mathcal{A}^*$ be the ρ_i-structure with the same universe as $\mathcal{A}$ and such that for each $R \in \rho$, $R^{\mathcal{A}^*} = R^{\mathcal{A}}$ and for all $V_j \in \vec{V}_i$, $V_j^{\mathcal{A}^*} = (\theta'_{i,j})^{\mathcal{A}}$. It follows from the definition of a substitution program that $\mathcal{A} \models \theta'_{n,i}[\alpha]$ if, and only if, $\mathcal{A}^* \models \theta_{n,i}[\alpha]$ if, and only if, $C_{n,i}[\gamma\mathcal{A}^*](\Omega_i(\gamma(\alpha(\vec{x}))))$. It thus suffices to show that $C_{n,i}[\gamma\mathcal{A}^*](\Omega_i(\gamma(\alpha(\vec{x})))) = C'_{n,i}[\gamma\mathcal{A}](\Omega'_i(\gamma(\alpha(\vec{x}))))$.

We prove by induction on the structure of $C_{n,i}$ that for each $g \in G_{n,i}$ if g is an input gate in $C_{n,i}$ labelled by V_j for some $j > i$ then $C_{n,i}[\gamma\mathcal{A}^*](g) = C'_{n,j}[\gamma\mathcal{A}](\Omega'_j((\Lambda_i)_{V_j}(g)))$ and otherwise $C_{n,i}[\gamma\mathcal{A}^*](g) = C'_{n,i}[\gamma\mathcal{A}](g)$. Let $g \in G_i$. Suppose g is an input gate. If g is labelled by V_j for some $j > i$ then $C_{n,i}[\gamma\mathcal{A}^*](g) = 1$ if, and only if, $\gamma^{-1}((\Lambda_i)_{V_j}(g)) \in V_j^{\mathcal{A}^*}$ if, and only if, $\mathcal{A} \models \theta'_{n,j}[\gamma^{-1}((\Lambda_i)_{V_j}(g))]$ if, and only if, $C'_{n,j}[\gamma\mathcal{A}](\Omega'_j((\Lambda_i)_{V_j}(g)))$. Otherwise, g is an input gate not labelled by V_j for any $j > i$, and, from the definition of $C'_{n,i}$, we have $C_{n,i}[\gamma\mathcal{A}^*](g) = C'_{n,i}[\gamma\mathcal{A}](g)$. Suppose g is not an input gate in $C_{n,i}$ and suppose the inductive hypothesis holds for each child of g in $C_{n,i}$. Let $a \in \mathrm{ind}(g)$. If $L_i(g)(a)$ is not an input gate labelled by V_j for any $j > i$ then, from the definition of $C'_{n,i}$ we have $L'_i(g)(a) = L_i(g)(a)$ and, from the induction hypothesis, $(L'_i)^{\gamma\mathcal{A}}(g)(a) = C'_{n,i}[\gamma\mathcal{A}](L'_i(g)(a)) =$

$C_{n,i}[\gamma\mathcal{A}^*](L_i(g)(a)) = L_i^{\gamma\mathcal{A}^*}(g)(a)$. Otherwise, $L_i(g)(a)$ is an input gate labelled by V_j for some $j > i$ and, from the definition of $C'_{n,i}$, we have $L'_i(g)(a) = \Omega'_j((\Lambda_i)_{V_j})(L_i(g)(a))$. It follows that that $L_i^{\mathcal{A}^*}(g)(a) = C_{n,i}[\gamma\mathcal{A}^*](L_i(g)(a)) = C'_{n,j}[\gamma\mathcal{A}](\Omega'_j((\Lambda_i)_{V_j})(L_i(g)(a))) = C'_{n,i}[\gamma\mathcal{A}](L'_i(g)(a)) = (L'_i)^{\gamma\mathcal{A}}(g)(a)$. The third equivalence follows from the definition of $C'_{n,i}$. It follows that $(L'_i)^{\gamma\mathcal{A}}(g) = L_i^{\mathcal{A}^*}(g)$, and so $C_{n,i}[\gamma\mathcal{A}^*](g) = C'_{n,i}[\gamma\mathcal{A}](g)$. This completes the inductive argument. From the definitions of the circuits $C_{n,i}$ and $C'_{n,i}$ we have that $\Omega'_i = \Omega_i$ and that no output gate in either circuit is also an input gate. In particular, we have $C'_{n,i}[\gamma\mathcal{A}](\Omega'_i(\gamma(\alpha(\vec{x})))) = C_{n,i}[\gamma\mathcal{A}^*](\Omega_i((\gamma\alpha(\vec{x}))))$. This completes the proof of Claim 5.3.3. □

Let $C'_n := C'_{n,1}$ and let θ'_n be the flattening of Θ'_n. It follows from the above three claims that C'_n translates θ'_n for n.

It remains to argue that the construction of C'_n for each n may be completed in time polynomial in n. This follows from three polynomial bounds. First, since Θ is P-uniform there exists a polynomial p_1 such that the function $n \mapsto \Theta_n$ is computable in time $p_1(n)$. Second, it follows from Lemma 5.1 that there is a polynomial p_2 such that the function that maps Θ_n to the sequence of circuits $C_{n,1}, \ldots, C_{n,|\Theta_n|}$ is computable in time $\sum_{i\in[|\Theta_n|]} p_2(|\theta_{n,i}|n^{\text{owidth}(\theta)+\text{width}(\theta)}) \leq p_1(n) \cdot p_2(p_1(n)n^c)$, where c is the constant such that for all $n \in \mathbb{N}$ and $i \in [|\Theta_n|]$, $\text{owidth}(\theta_{n,i}) \leq c$ and $\text{width}(\theta) \leq c$. Thirdly, the construction of C'_n from $C_{n,1}, \ldots, C_{n,|\Theta_n|}$ works by constructing each $C'_{n,i}$ from $C_{n,j}$, for all $j > i$, by replacing the appropriate input gates of $C_{n,i}$ with the output gates of $C'_{n,j}$. It can be shown that this algorithm runs in time polynomial in the combined size of these circuits. The function that maps $n \mapsto C'_n$ is the composition of these three functions, each of which is computable in time polynomial in n, and so $n \mapsto C'_n$ can be computed in time polynomial in n. This completes the proof of the proposition. □

Let $\mathbf{\Omega}$ be a set of almost relational generalised operators. There are actually two bases we define from $\mathbf{\Omega}$. The first is the basis $\mathbb{B}_{\mathbf{\Omega}}$ corresponding to $\mathbf{\Omega}$. The second is the basis $\mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$ corresponding to $\mathbf{Q}_{\mathbf{\Omega}}$, where $\mathbf{Q}_{\mathbf{\Omega}}$ is the set of quantifiers corresponding to $\mathbf{\Omega}$. It can be shown that $\mathbb{B}_{\mathbf{\Omega}} \subseteq \mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$, and that $\mathbb{B}_{\mathbf{\Omega}} = \mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$ only if every $\Omega \in \mathbf{\Omega}$ is relational and quantifies over the universe. As such, the basis $\mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$ is in general a richer basis then $\mathbb{B}_{\mathbf{\Omega}}$. We now show that this choice does not affect the power of the corresponding circuit models. In particular, we now show that there is an algorithm that takes as input a circuit C defined over the basis $\mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$ and outputs an equivalent circuit C' defined over the basis $\mathbb{B}_{\mathbf{\Omega}}$.

We define $\mathbf{Q}_{\mathbf{\Omega}}$ from $\mathbf{\Omega}$ by explicitly associating with each relation R in the vocabulary of $\mathbf{\Omega}$ a family of relations of the form $R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$, one for each possible assignment to the number variables bound in the definition of R (given by the arity of $\mathbf{\Omega}$). This induces a map between structures defined such that each tuple in R corresponds to a tuple in some appropriate relation $R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$. We prove the following result by reconstructing R from the associated family of relations. In other words, we replace each gate labelled by a function in $\mathbb{B}_{\mathbf{Q}_{\mathbf{\Omega}}}$ with

a gate labelled by a function in $\mathbb{B}_{\mathbf{\Omega}}$ and rewire the circuit so as to replace each wire labelled by a tuple in the relation $R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$ with an appropriate tuple in the relation R.

Lemma 5.4. *Let ρ be a vocabulary. Let $\mathbf{\Omega}$ be a set of almost relational generalised operators. Let $\mathbf{Q}$ be the corresponding set of quantifiers. Let $\mathbb{B}$ be any basis. There is an algorithm that takes as input a $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}, \rho)$-circuit C and outputs a $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}, \rho)$-circuit C' such that C and C' compute the same query and the algorithm runs in polynomial time. Moreover, if C is symmetric then C' is symmetric and if C is transparent then C' is transparent.*

Proof. We define $C' := \langle G', \Omega', \Sigma', \Lambda', L' \rangle$ as follows. Let $\mathbf{\Omega}^B$ be the set of Boolean-valued generalised operators corresponding to $\mathbf{\Omega}$. Let $G' := G$, $\Omega' := \Omega$, and $\Lambda' := \Lambda$. We define Σ' and L' as follows. Let $g \in G'$. If $\Sigma(g) \notin \mathbb{B}_{\mathbf{Q}}$ let $\Sigma'(g) = \Sigma(g)$ and let $L'(g) = L(g)$. Otherwise $\Sigma(g) \in \mathbb{B}_Q$ for some $Q \in \mathbf{Q}$. Let $\Omega \in \mathbf{\Omega}^B$ be such that $Q \in \mathbf{Q}_\Omega$. Let E be the evaluation function and ar be the arity of Ω. Let $\mathrm{ar}^1 : \boldsymbol{S} \to \mathbb{N}_0$ be defined such that $\mathrm{ar}^1(s) = \mathrm{ar}(s, 1)$ for all $s \in \boldsymbol{S}$. Let $\tau = (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be the vocabulary of Ω. Then $Q = Q_{E,\alpha,n,\mathrm{ar}}$ for some $\alpha : \boldsymbol{S} \to \mathbb{N}_0$ and $n \in \mathbb{N}$. Let $\tau_Q = (\boldsymbol{R}_Q, \boldsymbol{S}_Q, \zeta_Q)$ be the vocabulary of Q. It follows from the definition of $Q_{E,\alpha,n,\mathrm{ar}}$ that

$$\boldsymbol{R}_Q = \{R_{\vec{b}_1,\ldots,\vec{b}_{r_R}} : R \in \boldsymbol{R},\ \vec{b}_1 \in [n]_0^{\mathrm{ar}(\zeta(R)(1),2)}, \ldots, \vec{b}_{r_R} \in [n]_0^{\mathrm{ar}(\zeta(R)(r_R),2)}\}.$$

Let $X = \uplus_{s \in \boldsymbol{S}} X_s$ be the universe of $\Sigma(g)$. For each $s \in \boldsymbol{S}$ let $X'_s := X_s \times [n]^{\mathrm{ar}(s,2)}$ and let $X' = \uplus_{s \in \boldsymbol{S}} X'_s$. Let $\Sigma'(g) := F_{\Omega,\alpha}[X']$. Let $L'(g) : \tau[X'] \to G'$ be defined as follows. Let $(c, R) \in \tau[X']$. For each $i \in [r_R]$ there exists $(a_i, \vec{b}_i) \in X_{s_i} \times [n]^{\mathrm{ar}(s_i,2)}$ where $s_i = \zeta(R)(i)$ such that $c = ((a_1, \vec{b}_1), \ldots, (a_{r_R}, \vec{b}_{r_R}))$. Let $L'(g)(c, R) = L(g)((a_1, \ldots, a_{r_R}), R_{\vec{b}_1,\ldots,\vec{b}_{r_R}})$. It follows from the fact that $G' = G$, $\Lambda' = \Lambda$, and $\Omega' = \Omega$ that C' is a circuit.

Let k be the order of C. It follows that C' also has order k. We now show by induction that for $g \in G'$, $\mathcal{A} \in \mathrm{fin}[\rho, k]$, and $\gamma \in [k]^{\underline{A}}$ we have $C[\gamma\mathcal{A}](g) = C'[\gamma\mathcal{A}](g)$. The only interesting case in this inductive argument is when $\Sigma'(g) \in \mathbb{B}_{\mathbf{\Omega}}$. Then $\Sigma(g) \in \mathbb{B}_Q$ for some $Q \in \mathbf{Q}$. Let Ω, τ, τ_Q, E, ar X, and X' be defined as above. Let $\mathcal{G}$ be the class of structures associated with Q. It follows from the definition of a corresponding family of quantifiers that for any τ_Q-structure $\mathcal{B}$ with universe X we have $\mathcal{B} \in \mathcal{G}$ if, and only if, $E(\mathcal{B}^*) = 1$ where $\mathcal{B}^*$ is the τ-structure with universe X' and such that for each $F \in \boldsymbol{F}$, $F^{\mathcal{B}^*} = \alpha(F)$ and for all $R \in \boldsymbol{R}$, $i \in [r_R]$, and $(a_i, \vec{b}_i) \in X'_{\zeta(R)(i)}$ we have $((a_1, \vec{b}_1), \ldots, (a_{r_R}, \vec{b}_{r_R})) \in R^{\mathcal{B}^*}$ if, and only if, $(a_1, \ldots, a_{r_R}) \in R^{\mathcal{B}}_{\vec{b}_1,\ldots,\vec{b}_{r_R}}$. We note that from the induction hypothesis we have for all $R \in \boldsymbol{R}$, $i \in [r_R]$, and $(a_i, \vec{b}_i) \in X'_{\zeta(R)(i)}$ that

$$(L')^{\gamma\mathcal{A}}(g)(((a_1, \vec{b}_1), \ldots, (a_{r_R}, \vec{b}_{r_R})), R) = L^{\gamma\mathcal{A}}(g)((a_1, \ldots, a_{r_R}), R_{\vec{b}_1,\ldots,\vec{b}_{r_R}}).$$

We therefore have $C[\gamma\mathcal{A}](g) = 1$ if, and only if, $F_Q[X](L^{\gamma\mathcal{A}}(g)) = 1$ if, and only if, $L^{\gamma\mathcal{A}}(g) \in \mathcal{G}$ if, and only if, $E((L')^{\gamma\mathcal{A}}(g)) = 1$ if, and only if, $F_{\Omega,\alpha}[X']((L')^{\gamma\mathcal{A}}(g)) = 1$ if, and only if,

$C'[\gamma\mathcal{A}](g) = 1$. This concludes the inductive argument. Since $\Omega' = \Omega$ it follows that C and C' define the same query. Moreover, it can be shown by a trivial inductive argument that if C is transparent (resp. symmetric) then C' is transparent (resp. symmetric). We construct C' from C by relabelling the children of gates in the circuit, and it is not hard to show that this can be done in polynomial time.

□

We now prove the main theorem of this chapter establishing that each extension of a fixed-point logic by family of P-bounded almost relational generalised operators may be translated to an equivalent P-uniform family of transparent symmetric circuits defined over the corresponding basis. This result follows almost immediately from Lemmas 3.24, 3.26 5.4, and Proposition 5.3.

Theorem 5.5. *Let ρ is a vocabulary and let $\mathbf{\Omega}$ be a* P*-bounded set of almost relational generalised operators. Every query definable in* $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$ *is definable by a* P*-uniform family of transparent symmetric* $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{std}, \rho)$*-circuits* $(C_n)_{n\in\mathbb{N}}$*.*

Proof. Let $\theta(\vec{x}) \in \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})[\rho]$. From Lemma 3.24 there exists a P-uniform family of $\mathrm{FO}^{\mathbb{N}}(\mathbf{\Omega})[\rho]$-substitution programs $\Theta^1 := (\Theta^1_n)_{n\in\mathbb{N}}$ with constant length such that Θ^1 decides the same query as θ. Let $\mathbf{Q}$ be the set of quantifiers corresponding to $\mathbf{\Omega}$. From Lemma 3.26 there exists a P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}})[\rho]$-substitution programs $\Theta^2 := (\Theta^2_n)_{n\in\mathbb{N}}$ with constant width such that Θ^1 and Θ^2 define the same query. From Proposition 5.3 there is a P-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{Q}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuits $(C'_n)_{n\in\mathbb{N}}$ that decides the same query as Θ^2. From Lemma 5.4 there is a P-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuits $(C_n)_{n\in\mathbb{N}}$ that decides the same query as $(C'_n)_{n\in\mathbb{N}}$, and so the same query as Θ^2, Θ^1, and θ. The result follows. □

Chapter 6

The Support Theorem

In this chapter we discus a few structural properties of symmetric circuits and develop a theory of supports. We say a set $S \subseteq [n]$ is a support of a gate g in a circuit C of order n if any permutation in $\mathbf{Sym}_n$ that fixes S pointwise also fixes g. The supports of gate g encode information about the structure of the stabiliser group and orbit of g and, as shown in Chapter 7, are efficiently computable. Moreover, as shown in Lemma 8.3, the evaluation of g for a given input structure is entirely determined by how the elements of the input structure are mapped to a support of g.

In this chapter we show that each gate in a circuit can be associated with a unique *canonical support*. The main result, which we call *the support theorem*, establishes an upper bound on the size of the canonical support of any gate g in terms of the size of the orbits of g. It follows from this result that for a polynomial-size family of reduced injective symmetric circuits $(C_n)_{n\in\mathbb{N}}$ there is a constant bound on the size of the canonical support of any gate g in any circuit C_n. We show that each element of the universe of a gate can also be associated with a support and that the support theorem extends to these elements as well.

The support theorem proved in this chapter generalises a similar result proved by Anderson and Dawar [3] for symmetric circuits with trivially invariant gates. Anderson and Dawar's proof crucially relies on the fact that the stabiliser group of any gate is exactly equal to the setwise stabiliser group of its children, which holds only if the gates in question are all trivially invariant. We develop novel techniques in order to prove this result in the more general setting, where we allow for circuits defined over arbitrary bases. We also go beyond Anderson and Dawar in extending these results to elements of the universes of the gates in a circuit.

This chapter is organised as follows. In Section 6.1 we introduce basic group-theoretic language and define the notions of a support and supporting partition. We also define the notion of a *canonical* support and prove a few related results. In Section 6.2 we characterise the stabiliser groups of the gates in the circuit and establish a lower bound on the orbits of the gates. We use these results to prove the support theorem. In Section 6.3 we show that

the action of $\mathbf{Sym}_n$ on the gates of a circuit can be extended so as to define an action on the elements of the universes of the gates in the circuit. We also extend the support theorem to the elements of the universes of these gates.

6.1 Supports and Supporting Partitions

In this section we formally define the notion of a support and develop some surrounding theory. We also define the notion of a *supporting partition*. A supporting partition generalises the notion of a support by replacing the subset with a partition and the requirement to fix the support pointwise with a requirement to fix each component of the partition setwise. We show that each subgroup of $\mathbf{Sym}_n$ can be associated with a unique coarsest supporting partition, which we call the *canonical supporting partition*. The definition of a support is standard and the notions of a supporting partition and canonical supporting partition were introduced by Anderson and Dawar [3].

Definition 6.1. Let $G \leq \mathbf{Sym}_n$ and let $S \subseteq [n]$. Then S is a *support* for G if $\mathbf{Stab}_n(S) \leq G$.

Definition 6.2. Let $G \leq \mathbf{Sym}_n$ and $\mathcal{P}$ be a partition of $[n]$. Then $\mathcal{P}$ is a *supporting partition* for G if $\mathbf{Stab}_n(\mathcal{P}) \leq G$.

Notice that if $\mathcal{P}$ is a supporting partition for G and $P \in \mathcal{P}$ then $\mathbf{Stab}_n([n] \setminus P) \leq \mathbf{Stab}_n(\mathcal{P}) \leq G$, i.e. $[n] \setminus P$ is a support for G. Let $\mathcal{P}, \mathcal{P}'$ be partitions of $[n]$. We say that $\mathcal{P}'$ is *no finer* than $\mathcal{P}$ (and denote this by $\mathcal{P} \preceq \mathcal{P}'$) if for all $P \in \mathcal{P}$ there exists $P' \in \mathcal{P}'$ such that $P \subseteq P'$. Anderson and Dawar [3] define an operation $\mathcal{E}$ on pairs of partitions of $[n]$, where $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is the partition of $[n]$ consisting of the equivalence classes of the transitive closure of the relation $\sim$ on $[n]$ defined by $a \sim b$ if, and only if, there exists $P \in \mathcal{P} \cup \mathcal{P}'$ such that $a, b \in P$.

Anderson and Dawar show that $\mathcal{E}$ maps pairs of supporting partitions to supporting partitions and that $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is no finer than either $\mathcal{P}$ or $\mathcal{P}'$. In fact, $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is the finest partition that is no finer than either $\mathcal{P}$ or $\mathcal{P}'$ (i.e. $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is the least upper bound of $\mathcal{P}$ and $\mathcal{P}'$). We state this result formally in Proposition 6.3.

Proposition 6.3 ([3, Proposition 2])**.** *Let $G \leq \boldsymbol{Sym}_n$ be a group and let $\mathcal{P}$ and $\mathcal{P}'$ be supporting partitions of G. Then $\mathcal{P} \preceq \mathcal{E}(\mathcal{P}, \mathcal{P}')$ and $\mathcal{P}' \preceq \mathcal{E}(\mathcal{P}, \mathcal{P}')$, and $\mathcal{E}(\mathcal{P}, \mathcal{P}')$ is a supporting partition of G.*

Anderson and Dawar, using Proposition 6.3, show that every group $G \leq \mathbf{Sym}_n$ has a unique coarsest supporting partition. We call this partition the *canonical supporting partition*, and denote it by $\mathbf{SP}(G)$. We now define the notion of a *canonical support*.

Definition 6.4. Let $G \leq \mathbf{Sym}_n$. Let $\|\mathbf{SP}(G)\| = \min\{|[n] \setminus P| : P \in \mathbf{SP}(G)\}$. We say that G has *small support* if $\|\mathbf{SP}(G)\| < \frac{n}{2}$.

Note that if $G \leq \mathbf{Sym}_n$ has small support then there exists a unique $P \in \mathbf{SP}(G)$ such that $P > \frac{n}{2}$.

Definition 6.5. Let $G \leq \mathbf{Sym}_n$ such that G has small support. Let $\mathrm{sp}(G) = [n] \setminus P$, where P is the largest element of $\mathbf{SP}(G)$. We call $\mathrm{sp}(G)$ the *canonical support* of G.

Lemma 6.6. *Let $G_1, G_2 \leq \boldsymbol{Sym}_n$ with $G_1 \leq G_2$. Then $\boldsymbol{SP}(G_1) \preceq \boldsymbol{SP}(G_2)$*

Proof. We have that $\mathbf{Stab}_n(\mathbf{SP}(G_1)) \leq G_1 \leq G_2$, and so $\mathbf{SP}(G_1)$ supports G_2. Since $\mathbf{SP}(G_2)$ is the coarsest supporting partition of G_2 we have that $\mathbf{SP}(G_1) \preceq \mathbf{SP}(G_2)$. □

Lemma 6.7. *Let $G_1, G_2 \leq \boldsymbol{Sym}_n$ such that $G_1 \leq G_2$ and G_1 has small support. Then G_2 has small support and $sp(G_2) \subseteq sp(G_1)$.*

Proof. By Lemma 6.6, we have that $\mathbf{SP}(G_1) \preceq \mathbf{SP}(G_2)$. Let P_1 be the largest element of $\mathbf{SP}(G_1)$, and note that since G_1 has small support, $|P_1| > \frac{n}{2}$. Then there exists a (unique) $P_2 \in \mathbf{SP}(G_2)$ such that $P_1 \subseteq P_2$. Thus G_2 has small support, and $\mathrm{sp}(G_2) = [n] \setminus P_2 \subseteq [n] \setminus P_1 = \mathrm{sp}(G_1)$. □

Lemma 6.8. *Let $G, H, K \leq \boldsymbol{Sym}_n$ such that G has small support and $G = H \cap K$. Then H and K have small support and $sp(G) = sp(H) \cup sp(K)$.*

Proof. The fact that H and K have small support follows from Lemma 6.7.

Let $\mathcal{Q} := \{P_H \cap P_K : P_H \in \mathbf{SP}(H), P_K \in \mathbf{SP}(K) \text{ and } \exists P \in \mathbf{SP}(G), P \subseteq P_H \cap P_K\}$. We first show that $\mathcal{Q}$ is a supporting partition of G. We have that for any $P_H \cap P_K, P'_H \cap P'_K \in \mathcal{Q}$, $P_H \cap P_K \cap P'_H \cap P'_K \neq \emptyset$ if, and only if, $P_H = P'_H$ and $P_K = P'_K$ if, and only if, $P_H \cap P_K = P'_H \cap P'_K$. By Lemma 6.6, we have that $\mathbf{SP}(G) \preceq \mathbf{SP}(H)$ and $\mathbf{SP}(G) \preceq \mathbf{SP}(K)$. It follows that for each $P \in \mathbf{SP}(G)$ there exists $P_K \in \mathbf{SP}(K)$ and $P_H \in \mathbf{SP}(H)$ such that $P \subseteq P_H \cap P_K$. So, for each $a \in [n]$ there is $P_a \in \mathbf{SP}(G)$, $P_H \in \mathbf{SP}(H)$ and $P_K \in \mathbf{SP}(K)$ such that $a \in P_a \subseteq P_H \cap P_K$. It follows that $\mathcal{Q}$ is a partition.

Moreover, we note that $\mathbf{Stab}_n(\mathcal{Q}) \leq \mathbf{Stab}_n(\mathbf{SP}(H))$ and $\mathbf{Stab}_n(\mathcal{Q}) \leq \mathbf{Stab}_n(\mathbf{SP}(K))$, and thus $\mathbf{Stab}_n(\mathcal{Q}) \leq \mathbf{Stab}_n(\mathbf{SP}(H)) \cap \mathbf{Stab}_n(\mathbf{SP}(K)) \leq H \cap K = G$. It follows that $\mathcal{Q}$ is a supporting partition of G, and so by definition of the canonical supporting partition $\mathcal{Q} \preceq \mathbf{SP}(G)$. Since G has small support, there is a part P_G in $\mathbf{SP}(G)$ with $|P_G| > \frac{n}{2}$. Then there exists $P_H \in \mathbf{SP}(H)$ and $P_K \in \mathbf{SP}(K)$ such that $P_G \subseteq P_H \cap P_K$, and so $|P_H \cap P_K| > \frac{n}{2}$. Thus $P_H \cap P_K$ is the unique largest element in $\mathcal{Q}$. It follows from $\mathcal{Q} \preceq \mathbf{SP}(G)$ that $P_G \subseteq P_H \cap P_K \subseteq P_G$. □

We state the following two results proved by Anderson and Dawar [3].

Lemma 6.9. *Let $G \leq \boldsymbol{Sym}_n$ and $\sigma \in \boldsymbol{Sym}_n$ then $\sigma \boldsymbol{SP}(G) = \boldsymbol{SP}(\sigma G \sigma^{-1})$.*

Lemma 6.10. *For any $G \leq \boldsymbol{Sym}_n$ we have that $\boldsymbol{Stab}_n(\boldsymbol{SP}(G)) \leq G \leq \boldsymbol{SetStab}_n(\boldsymbol{SP}(G))$.*

6.1.1 Group Action on Supports

In our more general setting we are often interested not just in associating each gate in the circuit with a support but also in associating each element of the universe of a gate with a support. In this subsection we develop theory and terminology for dealing with group actions and supports with this more general application in mind.

Definition 6.11. Let X be a set on which a left group action of $G \leq \mathbf{Sym}_n$ is defined. We denote the *canonical supporting partition* of $x \in X$ by $\mathbf{SP}_G(x) = \mathbf{SP}(\mathbf{Stab}_G(x))$. Similarly we let $\|\mathbf{SP}_G(x)\| = \|\mathbf{SP}(\mathbf{Stab}_G(x))\|$. We say that $x \in X$ has *small support* if $\mathbf{Stab}_G(x)$ has small support. We say that X has *small supports* if each $x \in X$ has small support. If $x \in X$ has small support (i.e. $\|\mathbf{SP}_G(x)\| < \frac{n}{2}$), we denote the *canonical support* of x by $\mathrm{sp}_G(x) = \mathrm{sp}(\mathbf{Stab}_G(x))$. We refer to $\mathbf{SP}_G(x)$ (resp. $\mathrm{sp}_G(x)$) as the *canonical supporting partition* (resp. *canonical support*) of x *relative to* G.

If the subgroup $G \leq \mathbf{Sym}_n$ is obvious from context we omit the subscript in the canonical support and canonical support partition without the subscript. We have already noted that if C is a symmetric circuit with unique extensions of order n we can identify $\mathbf{Sym}_n$ with $\mathbf{Aut}(C)$ and so define an action of $\mathbf{Sym}_n$ on the gates of C. We omit the subscripts for the canonical support and canonical supporting partition when working with this group action.

We are often interested in supports relative to some stabiliser group and we now introduce some simplifying notation for that case. Let X and Y be sets on which a left group action of $\mathbf{Sym}_n$ is defined. Let $x \in X$ and $S \subseteq Y$. Let $\mathbf{SP}_S(x)$ and $\mathrm{sp}_S(x)$ abbreviate $\mathbf{SP}_{\mathbf{Stab}_n(S)}(x)$ and $\mathrm{sp}_{\mathbf{Stab}_n(S)}(x)$, respectively. Similarly, let $\mathbf{Orb}_S(x)$ and $\mathbf{Stab}_S(x)$ abbreviate $\mathbf{Orb}_{\mathbf{Stab}_n(S)}(x)$ and $\mathbf{Stab}_{\mathbf{Stab}_n(S)}(x)$, respectively. In the event that S is a singleton we omit the set braces in the subscript. We now extend a number of elementary results about supports to relative supports.

Lemma 6.12. *Let X be a set on which a left group action of $G \leq \boldsymbol{Sym}_n$ is defined and let $\sigma \in G$. Then for any $x \in X$, $\sigma \boldsymbol{Stab}_G(x)\sigma^{-1} = \boldsymbol{Stab}_G(\sigma x)$.*

Proof. Let $\pi \in \mathbf{Stab}_G(x)$, then $\sigma\pi\sigma^{-1}(\sigma x) = \sigma\pi x = \sigma x$, and so $\sigma\pi\sigma^{-1} \in \mathbf{Stab}_G(\sigma x)$. Let $\pi \in \mathbf{Stab}_G(\sigma x)$ then $\pi(\sigma x) = \sigma x$ and so $\sigma^{-1}\pi\sigma x = x$. It follows that $\sigma^{-1}\pi\sigma \in \mathbf{Stab}_G(x)$ and so $\pi = \sigma(\sigma^{-1}\pi\sigma)\sigma^{-1} \in \sigma\mathbf{Stab}_G(x)\sigma^{-1}$. □

Lemma 6.13. *Let X be a set on which a left group action of $G \leq \boldsymbol{Sym}_n$ is defined and let $\sigma \in G$. Then for any $x \in X$ it follows that $\sigma \boldsymbol{SP}_G(x) = \boldsymbol{SP}_G(\sigma x)$ and, if x has small support, $\sigma sp_G(x) = sp_G(\sigma x)$.*

Proof. We have Lemmas 6.9 and 6.12 that $\sigma\mathbf{SP}_G(x) = \sigma\mathbf{SP}(\mathbf{Stab}_G(x)) = \mathbf{SP}(\sigma\mathbf{Stab}_G(x)\sigma^{-1}) = \mathbf{SP}(\mathbf{Stab}_G(\sigma x)) = \mathbf{SP}_G(\sigma x)$, proving the first part of the statement.

From the fact that $\|\mathbf{SP}_G(x)\| < \frac{n}{2}$ it follows there exists a unique $P \in \mathbf{SP}_G(x)$ such that $|P| > \frac{n}{2}$ and $\mathrm{sp}_G(x) = [n] \setminus P$. But then $\sigma\mathrm{sp}_G(x) = \sigma([n] \setminus P) = [n] \setminus (\sigma P)$. We note that

$\sigma P \in \sigma\mathbf{SP}_G(x) = \mathbf{SP}_G(\sigma x)$ and $|\sigma P| > \frac{n}{2}$. Thus σP is the unique largest part in $\mathbf{SP}_G(\sigma x)$, and so $\mathrm{sp}_G(\sigma x) = [n] \setminus (\sigma P) = \sigma \mathrm{sp}_G(x)$. □

Lemma 6.14. *Let X be a set on which a left group action of $G \leq \boldsymbol{Sym}_n$ is defined. Let $x \in X$ be such that x has small support and let $\sigma, \sigma' \in G$. If $\sigma(a) = \sigma'(a)$ for all $a \in sp_G(x)$ then $\sigma(x) = \sigma'(x)$.*

Proof. We have that $(\sigma')^{-1}\sigma \in \mathbf{Stab}(\mathrm{sp}_G(x)) \subseteq \mathbf{Stab}_G(x)$ and so $(\sigma')^{-1}\sigma(x) = x$ and thus $\sigma(x) = \sigma'(x)$. □

6.2 The Support Theorem

The support theorem proved by Anderson and Dawar [3] establishes an upper bound on the size of the support of a gate in a symmetric circuit with trivially invariant gates in terms of sizes of the orbits of the gates. In particular, their result implies that if $(C_n)_{n\in\mathbb{N}}$ is a polynomial-size family of reduced symmetric circuits with majority gates then for large enough n every gate in C_n has a constant-size canonical support. In this section we generalise the support theorem and establish an analogous results for reduced injective symmetric circuits with no restriction on the basis.

The proof of the support theorem in this section follows a strategy broadly similar to the one used in [3], and uses two lemmas from there. The first of these lemmas gives us that if the index of a group $G \leq \mathbf{Sym}_n$ is small then $\mathbf{SP}(G)$ either has very few or very many parts. The second lemma gives us that for $G \leq \mathbf{Sym}_n$ of small index, if $\mathbf{SP}(G)$ has very few parts then it must have a single very large part (and hence a small canonical support). These two results allow us to conclude that a gate in a symmetric circuit has a small canonical support if it has a canonical supporting partition with very few parts. We prove by structural induction that the canonical supporting partition of every gate has few parts. To be precise, we show that if g is a minimal gate in the natural partial order on the circuit that has a canonical supporting partition with too many parts then the size of its orbit exceeds a given bound.

In order to prove this result we first characterise the stabiliser groups of the gates in a circuit. We then use this characterisation to derive a lower-bound on the size of the orbits of the gates. We recall that if C is a reduced symmetric circuit with trivially invariant gates then a permutation $\sigma \in \mathbf{Sym}_n$ moves g if, and only if, there exists $h \in H_g$ such that $\sigma h \notin H_g$. This simple observation is central to Anderson and Dawar's proof of the support theorem for circuits with trivially invariant gates, but it is false in our more general setting. To see how this can assertion can fail in this more general setting please see Example 6.15.

Example 6.15. Let $\tau = \{E\}$ be the graph vocabulary. Let $n \in \mathbb{N}$ and let $F : \{0,1\}^{\tau[n]} \to \{0,1\}$ be an isomorphism-invariant structured function. Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a circuit of order n that takes as input τ-structures defined as follows. Let $X = \{(i,j) \in [n]^2 : i \neq j\}$.

Let g_{out} be a gate, for each $i, j \in [n]$ let $g^{\text{in}}_{i,j}$ be a gate, and for each $(i,j) \in X$ let $g_{i,j}$ be a gate. Let $\Omega := g_{\text{out}}$. For each $(i,j) \in [n]^2$ let $\Sigma(g^{\text{in}}_{i,j}) := E$, for each $(i,j) \in X$ let $\Sigma(g_{i,j}) := F$, and let $\Sigma(g_{\text{out}}) := \text{AND}[X]$. For each $i, j \in [n]$ let $\Lambda(i,j) := g^{\text{in}}_{i,j}$. For each $(i,j) \in X$ let $L(g_{\text{out}})(i,j) := g_{i,j}$. For each $(i,j) \in X$ and each $(a,b) \in [n]^2$ let $L(g_{i,j})(a,b) := \Lambda(i,i)$ if $a = i$ and $b = j$, $L(g_{i,j})(a,b) := \Lambda(i,j)$ if $a = b = i$, and $L(g_{i,j})(a,b) := \Lambda(a,b)$ otherwise.

We should explain the intuition behind this formal definition. We think of C as having three layers. The first consists of all the relational gates. The second consists of the internal gates of the form $g_{i,j}$, one for each pair $(i,j) \in [n]^2$ such that $i \neq j$. We note that each of these gates has exactly the same set of children and has injective labels. Importantly, each $g_{i,j}$ labels its children slightly differently, labelling the relational gate $\Lambda(i,i)$ with the tuple (i,j), the gate $\Lambda(i,j)$ with the tuple (i,i), and labelling the rest of the gates in accord with Λ. The final layer of the gate contains a single output gate that takes a conjunction of all the gates in the second layer.

It is not hard to see that for any permutation $\sigma \in \mathbf{Sym}_n$ the bijection $\pi_\sigma : G \to G$ permutes the relational gates in accord with σ, maps each gate of the form $g_{i,j}$ to $g_{\sigma(i),\sigma(j)}$, and fixes the output gate, is a circuit automorphism extending σ. It is thus easy to see that C is a reduced symmetric circuit with injective labels. We also notice that for every permutation that moves any gate $g_{i,j}$ we have that $\sigma H_{g_{i,j}} = H_{g_{i,j}}$. In contrast, it was observed by Anderson and Dawar [3] that if a circuit is symmetric, reduced, and has only trivially invariant gates then a permutation σ moves a gate g if, and only if, $\sigma H_g \neq H_g$.

We now introduce the notion of a two gates being *compatible* with a permutation. We then show that a permutation $\sigma \in \mathbf{Sym}_n$ moves g if, and only if, two children of g are not compatible with σ.

Definition 6.16. Let C be an injective symmetric circuit of order n and let g be a gate in the circuit. Let $\sigma \in \mathbf{Sym}_n$ and $h, h' \in H_g$. We say that (h, h') is *compatible* with σ if $\sigma h, \sigma h' \in H_g$ and there is a partial automorphism λ on $\text{str}(g)$ such that $L(g)^{-1}(\sigma h) = \lambda(L(g)^{-1}(h))$ and $L(g)^{-1}(\sigma h') = \lambda(L(g)^{-1}(h'))$.

Lemma 6.17. *Let C be a reduced injective symmetric circuit of order n, g be a gate in the circuit, and $\sigma \in \boldsymbol{Sym}_n$. The following are equivalent*

1. *$\sigma H_g = H_g$ and $\sigma L(g)$ is isomorphic to $L(g)$,*
2. *$\sigma \in \boldsymbol{Stab}_n(g)$, and*
3. *for all $h, h' \in H_g$, (h, h') is compatible with σ.*

Proof. '1 $\Rightarrow$ 2': From the definition of a circuit automorphism we have that $L(\sigma g)$ and $\sigma L(g)$ are isomorphic, and so $L(\sigma g)$ and $L(g)$ are isomorphic. It follows that g and σg are syntactically-equivalent and so, since C is reduced, it follows that $g = \sigma g$.

'2 ⇒ 3' We have $\lambda \in \mathbf{Aut}(g)$ such that $L(g)\lambda = \sigma L(g)$. Then $\lambda = L(g)^{-1}\sigma L(g)$ and so $\lambda(L(g)^{-1}(h)) = L(g)^{-1}(\sigma L(g)(L(g)^{-1}(h))) = L(g)^{-1}(\sigma h)$ and similarly $\lambda(L(g)^{-1}(h')) = L(g)^{-1}(\sigma h')$.

'3 ⇒ 1' From the compatibility condition we have for all $h, h' \in H_g$ that $\sigma h, \sigma h' \in H_g$. It follows that $\sigma H_g = H_g$. We have for all $(\vec{a}_1, R_1), (\vec{a}_2, R_2) \in \mathrm{ind}(g)$ that there exists a partial automorphism λ' of $\mathrm{str}(g)$ such that $L(g)^{-1}(\sigma(L(g)(\vec{a}_1, R_1))) = \lambda'(\vec{a}_1, R_1)$ and $L(g)^{-1}(\sigma(L(g)(\vec{a}_2, R_2))) = \lambda'(\vec{a}_2, R_2)$. It follows that the function $\lambda : \mathrm{ind}(g) \to \mathrm{ind}(g)$ defined by $\lambda(\vec{a}, R) = L(g)^{-1}(\sigma(L(g)(\vec{a}, R)))$ for all $(\vec{a}, R) \in \mathrm{ind}(g)$ is in $\mathbf{Aut}(g)$. We then have that $\lambda L(g) = \sigma L(g)$ and so $\sigma L(g)$ is isomorphic to $L(g)$.

□

We aim to establish a lower bound on the sizes of the orbits of the gates in the circuit. We do this by establishing the existence of a large set of permutations that each take g to a different gate. To construct this set, we define a set of triples of the form (σ, h, h') where $\sigma \in \mathbf{Sym}_n$ and $h, h' \in H_g$. Each of these triples is useful (formally defined below) in a sense that guarantees that σ moves g. Moreover, the triples are pairwise independent which means that we can compose them in arbitrary combinations to generate new permutations moving g, while guaranteeing that each such combination gives us a different element in the orbit of g. We now define these terms formally and prove the result.

Definition 6.18. Let C be a circuit with injective labels of order n and g be a gate in C. We say that $(\sigma, h, h') \in \mathbf{Sym}_n \times H_g^2$ is *useful* if (h, h') is not compatible with σ.

The *mutual independence* relation on $\mathbf{Sym}_n \times H_g^2$ is defined to be the symmetric closure of the binary relation on $\mathbf{Sym}_n \times H_g^2$ consisting of all pairs $((\sigma_1, h_1, h_1'), (\sigma_2, h_2, h_2')) \in (\mathbf{Sym}_n \times H_g^2)^2$ such that

- $\sigma_2 h_1 = h_1$,
- $\sigma_2 \sigma_1 h_1 = \sigma_1 h_1$,
- $\sigma_2 h_1' = h_1'$, and
- $\sigma_2 \sigma_1 h_1' = \sigma_1 h_1'$.

If a pair of triples is in the mutual independence relation we say they are *mutually independent*. We say that a set $S \subseteq \mathbf{Sym}_n \times H_g^2$ is *useful* (at g) if each pair in it is useful. We say that S is *independent* (at g) if every two distinct elements in S are mutually independent.

Lemma 6.19. *Let C be a rigid injective symmetric circuit of order n and let g be a gate in that circuit. If S is a useful and independent set at g then $|\boldsymbol{Orb}(g)| \geq 2^{|S|}$.*

Proof. It follows from the orbit-stabiliser theorem that $|\mathbf{Orb}(g)| = [\mathbf{Sym}_n : \mathbf{Stab}_n(g)]$. We fix some linear order on S. For any $R \subseteq S$ define $\sigma_R = \Pi_{(\sigma,h,h') \in R}\sigma$, where the order of multiplication is induced by the order on S. We now show that the map $R \mapsto \sigma_R$ defines

a function from 2^S to $\mathbf{Sym}_n$ such that for all R and Q distinct subsets of S we have $\sigma_Q^{-1}\sigma_R \notin \mathbf{Stab}_n(g)$. We assume, without a loss of generality, that $R \setminus Q \neq \emptyset$. Pick any $(\sigma, h, h') \in R \setminus Q$. From independence we have $\sigma_R h = \sigma h$, $\sigma_R h' = \sigma h'$, $\sigma_Q \sigma h = \sigma h$, and $\sigma_Q \sigma h' = \sigma h'$. Thus $\sigma_Q^{-1}\sigma_R h = \sigma_Q^{-1}\sigma h = \sigma h$, and similarly $\sigma_Q^{-1}\sigma_R h' = \sigma h'$. Since S is useful it follows that (h, h') is incompatible with σ and so (h, h') is incompatible with $\sigma_Q^{-1}\sigma_R$. It follows from Lemma 6.17 that $\sigma_Q^{-1}\sigma_R \notin \mathbf{Stab}_n(g)$.

□

We earlier referred to two lemmas from [3] that we use to prove the support theorem. We now state these lemmas and explain their significance. Lemma 6.20 is used to establish a size bound on the supporting partition of a gate. In particular, it shows that for a partition $\mathcal{P}$ of $[n]$, if the index of $\mathbf{SetStab}(\mathcal{P})$ in $\mathbf{Sym}_n$ is small enough, then $\mathcal{P}$ either contains very few or very many parts.

Lemma 6.20 ([3, Lemma 5])**.** *For any ϵ and n such that $0 < \epsilon < 1$ and $\log n \geq \frac{4}{\epsilon}$, if $\mathcal{P}$ is a partition of $[n]$ with k parts, $s = [\boldsymbol{Sym}_n : \boldsymbol{SetStab}(\mathcal{P})]$ and $n \leq s \leq 2^{n^{1-\epsilon}}$, then $\min\{k, n-k\} \leq \frac{8}{\epsilon}\frac{\log s}{\log n}$.*

Lemma 6.21 gives us that, under the same assumptions as Lemma 6.20, if the number of parts in $\mathcal{P}$ is less then $\frac{n}{2}$ then $\mathcal{P}$ contains a very large part. This lemma is used both to establish that the stabiliser group of a gate has small support, and hence a canonical support, but also that this canonical support has bounded size (and is in fact constant for polynomial-size circuits).

Lemma 6.21 ([3, Lemma 6])**.** *For any ϵ and n such that $0 < \epsilon < 1$ and $\log n \geq \frac{8}{\epsilon^2}$, if $\mathcal{P}$ is a partition of $[n]$ with $|\mathcal{P}| \leq \frac{n}{2}$, $s := [\boldsymbol{Sym}_n : \boldsymbol{SetStab}(\mathcal{P})]$ and $n \leq s \leq 2^{n^{1-\epsilon}}$, then $\mathcal{P}$ contains a part P with at least $n - \frac{33}{\epsilon} \cdot \frac{\log s}{\log n}$ elements.*

We are now ready to prove the support theorem for symmetric circuits that may include non-trivially invariant gates. Let C be a circuit with unique extensions. We let $\mathbf{SP}(C)$ denote the maximum value of $\|\mathbf{SP}(g)\|$ for g a gate in C.

Theorem 6.22. *For any ϵ and n such that $\frac{2}{3} \leq \epsilon \leq 1$ and $n \geq \frac{128}{\epsilon^2}$, if C is a reduced injective symmetric circuit of order n and $s := \max_{g \in C} |\boldsymbol{Orb}(g)| \leq 2^{n^{1-\epsilon}}$, then, $\boldsymbol{SP}(C) \leq \frac{33}{\epsilon}\frac{\log s}{\log n}$.*

Proof. First we note that if $1 \leq s < n$, then C cannot have a relational gate, as the orbit of a relational gate has at least n elements. Since C has no relational gates, the only input gates in the circuit are the constant gates. Since constant gates are fixed by all permutations, it follows that any gate g whose children are constant gates must similarly be fixed under all permutations. Furthermore, from the fact that C is reduced (so from Proposition 4.18 has unique extensions) this property inductively extends to the rest of the circuit. Thus for each gate g in C the partition $\{[n]\}$ supports g, and since this is trivially the coarsest such partition it follows that $\|\mathbf{SP}(g)\| = 0 = \mathbf{SP}(C)$. We therefore assume that $s \geq n$.

If g is a gate in C then $\mathbf{Stab}(g) \leq \mathbf{SetStab}(\mathbf{SP}(g))$, and so $s \geq |\mathbf{Orb}(g)| = [\mathbf{Sym}_n : \mathbf{Stab}(g)] \geq [\mathbf{Sym}_n : \mathbf{SetStab}(\mathbf{SP}(g))]$. Thus if $|\mathbf{SP}(g)| \leq \frac{n}{2}$, then from Lemma 6.21, we have $\|\mathbf{SP}(g)\| \leq \frac{33}{\epsilon} \cdot \frac{\log s}{\log n}$. The result thus follows from showing that for each g in C we have that $|\mathbf{SP}(g)| \leq \frac{n}{2}$.

If g is a constant gate, then as argued above, it follows $|\mathbf{SP}(g)| = 0 < \frac{n}{2}$. If g is a relational gate, then g is fixed by a permutation $\sigma \in \mathbf{Sym}_n$ if, and only if, σ fixes all elements that appear in $\Lambda(g)$. It follows that $\{a\} \in \mathbf{SP}(g)$ for each a appearing in $\Lambda(g)$ and all other elements of $[n]$ are contained in a single part of $\mathbf{SP}(g)$. But suppose $|\mathbf{SP}(g)| > \frac{n}{2}$. Then the number of singletons in $\mathbf{SP}(g)$ must be larger than $\frac{n}{2}$, which in turn, from the orbit-stabiliser theorem, gives us that $s \geq |\mathbf{Orb}(g)| \geq \frac{n!}{(n-|\Lambda(g)|)!} \geq \frac{n!}{(\frac{n}{2})!} \geq 2^{\frac{n}{4}} > 2^{n^{1-\epsilon}}$. This is a contradiction, and so $|\mathbf{SP}(g)| \leq \frac{n}{2}$.

We now consider the internal gate case. Let g be the topologically first internal gate with $|\mathbf{SP}(g)| > \frac{n}{2}$. Let $k' := \lceil \frac{8 \log s}{\epsilon \log n} \rceil$. From the assumptions on s, n and ϵ we have that $k' \leq \frac{1}{4} n^{1-\epsilon} < \frac{n}{2}$. We note that Lemma 6.20 implies that $n - |\mathbf{SP}(g)| \leq k'$.

We now construct a sufficiently large useful and independent set of triples which, using Lemma 6.19, allows us to place a lower bound on the orbit size of g. Divide $[n]$ into $\lfloor \frac{n}{k'+2} \rfloor$ disjoint sets S_i of size $k'+2$ and ignore the elements left over. It follows that for each i there is a permutation σ_i which fixes $[n] \setminus S_i$ pointwise but moves g. Suppose there was no such σ_i, but then every permutation that fixes $[n] \setminus S_i$ pointwise fixes g. Thus the partition of all the singletons in $[n] \setminus S_i$ and S_i is a supporting partition of g. As $\mathbf{SP}(g)$ is the coarsest such partition it follows that $|\mathbf{SP}(g)| \leq n - (k'+2) + 1 = n - k' - 1$, which contradicts the inequality $n - |\mathbf{SP}(g)| \leq k'$.

Since g is moved by each σ_i, and C is reduced and injective, it follows from Lemma 6.17 that there exists $(h_i, h_i') \in H_g$ such that (h_i, h_i') is not compatible with σ_i, and so the triple (σ_i, h_i, h_i') is useful.

Note that our choice of g guarantees that for all $h \in H_g$, $|\mathbf{SP}(h)| \leq \frac{n}{2}$, and so, from Lemma 6.21 and the hypothesis of this theorem, h has small support. Let $Q_i = \mathrm{sp}(h_i) \cup \mathrm{sp}(\sigma_i h_i) \cup \mathrm{sp}(h_i') \cup \mathrm{sp}(\sigma_i h_i')$. Then note that if σ_j fixes Q_i pointwise then by construction we have that $\sigma_j \in \mathbf{Stab}(\mathbf{SP}(h_i)) \cap \mathbf{Stab}(\mathbf{SP}(\sigma_i h_i)) \cap \mathbf{Stab}(\mathbf{SP}(h_i')) \cap \mathbf{Stab}(\mathbf{SP}(\sigma_i h_i'))$

Define a directed graph K with vertices given by the sets S_i and an edge from S_i to S_j (with $i \neq j$) if, and only if, $Q_i \cap S_j \neq \emptyset$. It follows then that if there is no edge from S_i to S_j then $Q_i \subseteq [n] \setminus S_j$, and so σ_j fixes Q_i pointwise, giving us that (σ_i, h_i, h_i') and (σ_j, h_j, h_j') are mutually independent. It remains to argue that K has a large independent set. This is possible as the out-degree of S_i in K is bounded by

$$|Q_i| \leq \|\mathbf{SP}(h_i)\| + \|\mathbf{SP}(\sigma_i h_i)\| + \|\mathbf{SP}(h_i')\| + \|\mathbf{SP}(\sigma_i h_i') \leq 4 \cdot \frac{33 \log s}{\epsilon \log n}.$$

These inequalities follow from the fact that the sets S_i are disjoint and we may apply Lemma 6.21 to each of the child gates. From these inequalities we have that the average total degree

(in + out degree) of K is at most $2 \cdot |Q_i| \leq 34 \cdot k'$. Now greedily select a maximal independent set in K by repeatedly selecting S_i with the lowest total degree and eliminating it and its neighbours. This action does not affect the bound on the average total degree of K and hence determines an independent set I in K of size at least

$$\frac{\lfloor \frac{n}{k'+2} \rfloor}{34k'+1} \geq \frac{n-(k'+2)}{34k'+1k'+2} \geq \frac{n\frac{7}{16}}{34k'^2+69k'+2} \geq \frac{n}{(16k')^2}.$$

Take $S = \{(\sigma_i, h_i, h_i') : S_i \in I\}$. Then from the above argument we have that S is useful and independent.

Moreover, from Lemma 6.19, we have that $s \geq |\mathbf{Orb}(g)| \geq 2^{|S|} \geq 2^{\frac{n}{(16k')^2}}$ then $n^{1-\epsilon} \geq \log s \geq n \cdot (\frac{128}{\epsilon}\frac{\log s}{\log n})^{-2} > n \cdot (n^{1-\epsilon})^{-2} = n^{2\epsilon-1} \geq n^{1-\epsilon}$. This is a contradiction, and the result follows. □

Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ be a polynomial-size family of reduced injective symmetric circuits. Then $s(n) := \max_{g\in C_n} |\mathbf{Orb}(g)|$ must be polynomially bounded, and so the theorem implies that there exists $k \in \mathbb{N}$ such that for all large enough n and all $g \in C_n$ we have that g has small support and $|\mathrm{sp}(g)| \leq k$.

Corollary 6.23. *Let $\mathcal{C} := (C_n)_{n\in\mathbb{N}}$ be a polynomial-size family of reduced injective symmetric circuits. There is a k such that $\boldsymbol{SP}(C_n) \leq k$ for all n.*

6.3 Supports on Indexes

We have defined an action of $\mathbf{Sym}_n$ on the gates of a circuit of order n and established a bound on the size of the supports of these gates. We now show that we can also define a natural action on the elements of the universes of the gates and that the support theorem can be extended so as to give a bound on the supports of these elements.

Let C be a reduced injective symmetric circuit of order n and let g be a gate in C. Since C is symmetric and reduced it follows from Proposition 4.18 that each permutation in $\mathbf{Sym}_n$ extends uniquely to an automorphism of C. We define a group action of $\mathbf{Stab}(g)$ on $\mathrm{unv}(g)$ as follows. Let $\sigma \in \mathbf{Stab}(g)$ and $x \in \mathrm{unv}(g)$. It follows from the fact that C is injective that there exists a unique $\lambda \in \mathbf{Aut}(g)$ such that $\lambda L(g) = L(\sigma g)$. Let $\sigma \cdot x = \lambda(x)$.

In this case we are considering the group action of $\mathbf{Stab}(g)$, rather than $\mathbf{Sym}_n$, and so we consider supports and stabilisers relative to $\mathbf{Stab}(g)$ or a subgroup of $\mathbf{Stab}(g)$. In order to simplify notation we use the abbreviations introduced in Section 6.1.1 for relative supports and stabilisers. We now prove an extension of the support theorem and show that for any polynomial-size family $(C_n)_{n\in\mathbb{N}}$ of reduced injective symmetric circuit we have for large enough n that for each g in C_n the supports of each $x \in \mathrm{unv}(g)$ relative to both the subgroups $\mathbf{Stab}(g)$ and $\mathbf{Stab}(\mathrm{sp}(g))$ have size bounded by a constant. The proof of this result follows almost immediately from the support theorem and the observation that for any

internal gate g the support of an element a in the universe of g relative to the support of g is always contained in the union of the support of g and the support of some child gate of g.

Theorem 6.24. *Let $(C_n)_{n\in\mathbb{N}}$ be a polynomial-size family of injective reduced symmetric circuits. There exists $n_0, k \in \mathbb{N}$ such that for all $n > n_0$, g a gate in C_n, and $a \in unv(g)$*

- $\boldsymbol{Stab}_n(g)$, $\boldsymbol{Stab}_{sp(g)}(a)$, *and* $\boldsymbol{Stab}_g(a)$ *have small supports,*
- *if $h \in H_g$ and a appears in $L(g)^{-1}(h)$ then $sp_g(a) \subseteq sp_{sp(g)}(a) \subseteq sp(g) \cup sp(h)$, and*
- $|sp(g)| \leq k$ *and* $|sp_g(a)| \leq |sp_{sp(g)}(a)| \leq 2k$.

Proof. We have from Corollary 6.23 that there exists $k, n_0' \in \mathbb{N}$ such that for all $n \geq n_0'$, $\mathbf{SP}(C_n) \leq k$. Let $n_0 = \max(n_0', 4k+1)$. Let $n > n_0$, g be a gate in C_n, $a \in \mathrm{unv}(g)$, and $h \in H_g$ be such that there exists $(\vec{a}, R) \in \mathrm{ind}(g)$ such that $(\vec{a}, R) = L(g)^{-1}(h)$ and $a \in \vec{a}$.

Since $\mathbf{SP}(C_n) \leq k$ and $n > 4k$, we have that $\mathbf{Stab}(g)$ and $\mathbf{Stab}(h)$ have small supports. Moreover, we have that $\mathbf{Stab}(\mathrm{sp}(h) \cup \mathrm{sp}(g)) = \mathbf{Stab}(\mathrm{sp}(h)) \cap \mathbf{Stab}(\mathrm{sp}(g)) \leq \mathbf{Stab}(h) \cap \mathbf{Stab}(\mathrm{sp}(g)) = \mathbf{Stab}_{\mathrm{sp}(g)}(h)$. It follows that $\mathrm{sp}(h) \cup \mathrm{sp}(g)$ supports $\mathbf{Stab}_{\mathrm{sp}(g)}(h)$ and so, since $|\mathrm{sp}(h) \cup \mathrm{sp}(g)| \leq 2k < \frac{n}{2}$, $\mathbf{Stab}_{\mathrm{sp}(g)}(h)$ has small support and $\mathrm{sp}_{\mathrm{sp}(g)}(h) \subseteq \mathrm{sp}(h) \cup \mathrm{sp}(g)$.

We also have that $\mathbf{Stab}_{\mathrm{sp}(g)}(h) = \bigcap_{i\in[|\vec{a}|]} \mathbf{Stab}_{\mathrm{sp}(g)}(a_i)$, and so by repeated application of Lemma 6.8, we have that $\mathrm{sp}_{\mathrm{sp}(g)}(a) \subseteq \bigcup_{i\in[|\vec{a}|]} \mathrm{sp}_{\mathrm{sp}(g)}(a_i) = \mathrm{sp}_{\mathrm{sp}(g)}(h) \subseteq \mathrm{sp}(h) \cup \mathrm{sp}(g)$. We thus also have that $|\mathrm{sp}_{\mathrm{sp}(g)}(a)| \leq 2k$.

Lastly, we note that $\mathbf{Stab}_{\mathrm{sp}(g)}(a) \subseteq \mathbf{Stab}_g(a)$, and so $\mathbf{Stab}_g(a)$ has small support and $\mathrm{sp}_g(a) \subseteq \mathrm{sp}_{\mathrm{sp}(g)}(a)$. The result follows. □

Chapter 7

Transparent Circuits

In order to establish our translation from P-uniform families of circuits to formulas of fixed-point logic we first need to show that a number of circuit parameters, such as the action of an automorphism or the syntactic-equivalence relation, are polynomial-time computable. Moreover, in order to apply the support theorem we need to be able to restrict our attention to reduced injective circuits, and for this we seem to need a polynomial-time computable translation from circuits to equivalent reduced injective circuits. However, computing these properties and translations seems to entail showing either that a given function is a valid circuit automorphism or that a given pair of gates are syntactically-equivalent. Both of these reduce to checking if two structures labelling gates in the circuit are isomorphic. If we restrict our attention to circuits with trivially invariant gates, as they do in [3], this isomorphism problem reduces to testing for the existence of a bijection, which is polynomial-time decidable. However, in our more general setting, where the gates of the circuit may be labelled by arbitrary relational structures, this isomorphism problem is in general as hard as the graph-isomorphism problem.

In order to ensure the polynomial-time decidability of these important problems we have restricted our attention to transparent circuits. We recall that a transparent circuit is one where every non-trivially invariant gate has unique labels. In this case there can be at most one easily identifiable label-preserving function between the structures labelling any two gates, and so deciding isomorphism reduces to deciding if this particular function is an isomorphism. This is a polynomial-time decidable problem.

This chapter is organised as follows. In Section 7.1 we formally show that the requisite circuit properties are indeed polynomial-time decidable for transparent circuits. We first show that the syntactic-equivalence relation for transparent circuits is polynomial-time decidable. We then establish a translation from transparent circuits to reduced injective circuits. We also show that other related properties, such as a circuit being symmetric, are polynomial-time decidable. In Section 7.2 we discuss more formally the relationship between the graph-isomorphism problem and the aforementioned decision problems. In each case we

show that the graph-isomorphism problem is reducible to deciding the given circuit property for general symmetric circuits. It is easy to establish the reduction in the reverse direction, and so it follows that each of these properties is polynomial-time decidable if, and only if, the graph-isomorphism problem is in P. We also show that most of these hardness results are robust in the sense that even if we restrict our attention to other natural classes of circuits (e.g. circuits with unique children or circuits with injective labels) these problems remain at least as hard as the graph-isomorphism problem.

We think of Section 7.1 as presenting a number of useful results for transparent circuits, and hence suggesting why the restriction to transparent circuits is useful. We think of Section 7.2 as proving a cumulative case for the necessity of this restriction.

7.1 Tractable Properties of Transparent Circuits

In this section we show that many important circuit properties are polynomial-time decidable for transparent circuits. First, we prove that the syntactic-equivalence relation is polynomial-time computable for transparent circuits. We use this result to show that transparent circuits can be transformed in polynomial-time into equivalent reduced injective circuits. We show that for circuits with unique labels we can compute in polynomial-time the action of a given automorphism on the circuit, the orbits and canonical supporting partitions of each gate, as well as the orbits and supporting partitions of each element of the universe of each gate. We use these results when we define our translation from circuit families to fixed-point formulas in Chapter 8.

We now show that the syntactic-equivalence relation can be computed in polynomial-time for transparent circuits.

Lemma 7.1. *There is an algorithm that takes as input a transparent circuit C and outputs the syntactic-equivalence relation on the gates of C. The algorithm runs in time polynomial in the size of C.*

Proof. Let $C := \langle G, \Omega, \Sigma, \Lambda, L \rangle$ be a transparent $(\mathbb{B}, \rho)$-circuit of order n. We should note that syntactic-equivilence is defined inductively, and this inductive definition can be implemented as a dynamic program. In order to show this algorithm runs in polynomial-time it suffices to show there is a polynomial-time algorithm that takes as input two gates g and h in C and the syntactic-equivalence relation for all gates of depth less than either g or h and returns whether or not $g \equiv h$. We now sketch a definition of this algorithm.

Let g and h be two internal gates in C and suppose we have defined the syntactic-equivalence relation for all gates of depth less than the depth of either g or h. We check all of the conditions for syntactic-equivalence with the exception of the isomorphism condition, and if any of them fail to hold we halt and output that $g \not\equiv h$.

Let $x_1, \ldots, x_{|\mathrm{ind}(g)|}$ be an indexing of the elements in $\mathrm{ind}(g)$ by $[|\mathrm{ind}(g)|]$. We iterate over each $i \in [|\mathrm{ind}(g)|]$ and select $y_i \in \mathrm{ind}(h)$ such that $y_i \neq y_j$ for all $j \in [i-1]$ and $L(g)(x_i) \equiv L(h)(y_i)$. If at any point we cannot select an appropriate y_i we halt and output that $g \not\equiv h$. For each $i \in [|\mathrm{ind}(g)|]$ let $f(x_i) = y_i$. We have that f is injective. Check if f is surjective. If not, halt and output that $g \not\equiv h$. We thus have that f is a bijective function and for all $x \in \mathrm{ind}(g)$, $L(g)(x) \equiv L(h)(f(x))$. Check if g and h are both trivially invariant gates. If so, f is an isomorphism from $L(g)/_{\equiv}$ to $L(h)/_{\equiv}$, and so we halt and output that $g \equiv h$.

We thus have that g and h are non-trivially invariant gates. Then, since C is transparent, both g and h have unique labels. In that case the function f is the only function such that for all $x \in \mathrm{ind}(g)$, $L(g)(x) \equiv L(h)(f(x))$. It follows that $g \equiv h$ if, and only if, f defines an isomorphism from $\mathrm{str}(g)$ to $\mathrm{str}(h)$. This is easy to check.

□

It is natural to consider quotients of algebraic structures by congruences. We now define what it means to take a quotient of a circuit by the syntactic equivalence relation. Intuitively, we think of a quotient of a given circuit as being formed by "merging" each syntactic-equivalence class into a single gate and including a wire between any two gates $[h]$ and $[g]$ in the quotient circuit if, and only if, for some $h' \in [h]$ and $g' \in [g]$ there is a wire from h' to g'.

Definition 7.2. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit. A *quotient of* C is a $(\mathbb{B}, \rho)$-circuit $C_{\equiv} := \langle G_{\equiv}, \Omega_{\equiv}, \Sigma_{\equiv}, \Lambda_{\equiv}, L_{\equiv}\rangle$, where $G_{\equiv} = G/_{\equiv}$, $\Omega_{\equiv} = \Omega/_{\equiv}$, $\Sigma = \Sigma/_{\equiv}$, $(\Lambda_{\equiv})_R = \Lambda_R/_{\equiv}$ for all $R \in \rho$, and for all $[g] \in G_{\equiv}$ there exists $g' \in [g]$ such that, $L_{\equiv}$ associates with $[g]$ a function $L_{\equiv}([g]) : \mathrm{ind}(g') \to G_{\equiv}$ where $L_{\equiv}([g]) = L(g')/_{\equiv}$.

We should note that the there is no obvious quotienting operation that associates with each circuit a unique quotient circuit. However, it is easy to see that if C and C' are distinct quotients of a given circuit then the only point where they differ is in the definition of their respective labelling functions. But it is not hard to see that for every gate g in C, $L(g)$ and $L'(g)$ are isomorphic. It follows that these two circuits are isomorphic in the precise sense alluded to right after Definition 4.9.

We now show that taking the quotient of a circuit preserves important properties, including the function computed by the circuit, the symmetry of the circuit, and whether the circuit has unique labels. We also show that the quotient of a circuit is reduced, and hence has unique children (and so, from Proposition 4.18, unique extensions).

Lemma 7.3. *Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit and $C_{\equiv} = \langle G_{\equiv}, \Omega_{\equiv}, \Sigma_{\equiv}, \Lambda_{\equiv}, L_{\equiv}\rangle$ be a quotient of C. Then $C_{\equiv}$ is reduced and C and $C_{\equiv}$ compute the same function. Moreover, if C is symmetric then $C_{\equiv}$ is symmetric, and for all $g \in G$, g has unique labels in C if, and only if, $[g]$ has unique labels in $C_{\equiv}$. Indeed, for all $\sigma \in \boldsymbol{Sym}_n$ if there exists $\pi \in \boldsymbol{Aut}(C)$ extending σ then $\pi/_{\equiv}$ is an automorphism of $C_{\equiv}$ extending σ.*

Proof. Let n be the order of C. We first prove that $C_{\equiv}$ and C compute the same function. Let $\mathcal{A}$ be a ρ-structure of size n and let γ be a bijection from A to $[n]$. We now show that for all $g \in G$, $C_{\equiv}[\gamma\mathcal{A}]([g]) = C[\gamma\mathcal{A}](g)$. We do this by induction on the depth of a gate. Suppose $g \in G$ has depth 0. In this case g is an input gate and the result follows trivially. Suppose g is an internal gate and suppose for all h of depth less than g we have that $C_{\equiv}[\gamma\mathcal{A}]([h]) = C[\gamma\mathcal{A}](h)$. We have that there exists $g' \in [g]$ such that $L_{\equiv}([g]) = L(g')/_{\equiv}$ and $\Sigma_{\equiv}([g]) = \Sigma(g) = \Sigma(g')$. We have from Lemma 4.13 and the inductive hypothesis that there exists $\lambda \in \mathbf{Aut}(g)$ such that $L_{\equiv}([g]) = L(g') = L(g)\lambda$. It follows from the fact that $\Sigma(g)$ is a structured function that $C_{\equiv}[\gamma\mathcal{A}]([g]) = \Sigma_{\equiv}([g])(L_{\equiv}^{\gamma\mathcal{A}}([g])) = \Sigma(g')(L^{\gamma\mathcal{A}}(g')) = \Sigma(g)(L^{\gamma\mathcal{A}}(g)\lambda) = \Sigma(g)(L^{\gamma\mathcal{A}}(g)) = C[\gamma\mathcal{A}](g)$.

We now show that $C_{\equiv}$ is reduced. Suppose $[g], [h] \in G_{\equiv}$ and suppose $[g] \equiv [h]$. If $[g]$ and $[h]$ are both input or output gates then $[g] = [h]$. Suppose $[g]$ and $[h]$ are internal gates that are not output gates. Then $\Sigma(g) = \Sigma_{\equiv}([g]) = \Sigma_{\equiv}([h]) = \Sigma(h)$. There exists $g', h' \in G$ such that $g' \equiv g$ and $h' \equiv h$, and $L_{\equiv}([g]) = L(g')/_{\equiv}$ and $L_{\equiv}([h]) = L(h')/_{\equiv}$. It follows from $[g] \equiv [h]$ that $L(g')/_{\equiv}$ is isomorphic to $L(h')/_{\equiv}$. From this it follows that $g' \equiv h'$, and so $[g] = [h]$.

Let $\sigma \in \mathbf{Sym}_n$ and suppose there exists $\pi \in \mathbf{Aut}(C)$ extending σ. Let $\pi_{\equiv} = \pi/_{\equiv}$. We now show that $\pi_{\equiv}$ is an automorphism of $C_{\equiv}$ extending σ. It is easy to see that $\pi_{\equiv}$ is a bijection from $G_{\equiv}$ to $G_{\equiv}$ that preserves syntactic-equivalence, and is thus is a well-defined function. Let $[g] \in G_{\equiv}$. We have that $\Sigma_{\equiv}(\pi_{\equiv}[g]) = \Sigma_{\equiv}([\pi g]) = \Sigma(\pi g) = \Sigma(g) = \Sigma_{\equiv}([g])$. It is easy to check the automorphism conditions for input gates. Suppose $[g]$ is an internal gate. Let $g' \in [g]$ be such that $L_{\equiv}([g]) = L(g')/_{\equiv}$ and $h \in [g]$ be such that $L_{\equiv}(\pi_{\equiv}[g]) = L_{\equiv}([\pi h]) = L(\pi h)/_{\equiv}$. We have that $\pi L(g')$ is isomorphic to $L(\pi g')$, and it follows that $(\pi L(g'))/_{\equiv}$ is isomorphic to $L(\pi g')/_{\equiv}$. We then have $(\pi L(g'))/_{\equiv} = \pi_{\equiv}(L(g')/_{\equiv}) = \pi_{\equiv}L_{\equiv}([g])$ and, since $g' \equiv h$ and so $\pi g' \equiv \pi h$, we have that $L(\pi g')/_{\equiv}$ is isomorphic to $L(\pi h)/_{\equiv} = L_{\equiv}(\pi_{\equiv}[g])$. It follows that $\pi_{\equiv}L_{\equiv}([g])$ is isomorphic to $L_{\equiv}(\pi_{\equiv}[g])/_{\equiv}$. Suppose $[g]$ is an output gate. Then for $\vec{a} \in \mathbf{Dom}(\Omega)$, $\pi_{\equiv}\Omega_{\equiv}(\vec{a}) = [\pi\Omega(\vec{a})] = [\Omega(\sigma\vec{a})] = \Omega_{\equiv}(\sigma\vec{a})$. It follows that if C is symmetric then $C_{\equiv}$ is symmetric.

Let $g \in G$. Suppose $[g]$ has unique labels. There exists $h \in [g]$ such that $L_{\equiv}([g]) = L(h)/_{\equiv}$. Since $L_{\equiv}([g])$ is injective, $L(h)/_{\equiv}$ must be injective and so $L(h)$ must be injective and no two child gates of h can be syntactically-equivalent. It follows that h has unique labels. Since $h \equiv g$ and h has unique labels, it follows that g has unique labels. Suppose g has unique labels. Let $h \in [g]$ be such that $L_{\equiv}([g]) = L(h)/_{\equiv}$. Since $h \equiv g$ and g has unique labels h has unique labels and so $L_{\equiv}([g])$ has injective labels. Since each equivalence class in $C_{\equiv}$ is a singleton it follows that $[g]$ has unique labels. □

In is not hard to show that there is a polynomial-time computable function that maps a transparent circuit to a quotient of that circuit. To see this, recall that from Lemma 7.1 we can compute the syntactic-equivalence classes of a transparent circuit in polynomial-time. We

can thus define a quotient circuit by picking representatives from each syntactic-equivalence class and then applying the definition of a quotient circuit in the obvious manner.

We now show that we can transform in polynomial time a transparent circuit into an equivalent reduced injective circuit, and hence one with unique labels and unique extensions, and that this transformation preserves important properties such as symmetry.

Lemma 7.4. *There is an algorithm that takes as input a transparent $(\mathbb{B}, \rho)$-circuit C and outputs a $(\mathbb{B} \cup \mathbb{B}_{\boldsymbol{std}}, \rho)$-circuit C' such that C and C' compute the same function, C' is reduced and injective, and if C is symmetric then C' is symmetric. Moreover, this algorithm runs in time polynomial in the size of the input circuit.*

Proof. Let $\langle G, \Omega, \Sigma, \Lambda, L\rangle := C$. Let $C_0 := \langle G_0, \Omega_0, \Sigma_0, \Lambda_0, L_0\rangle$ be the quotient of C. If C_0 does not contain the constant gates g_0 and g_1 such that $\Sigma(g_0) = 0$ and $\Sigma(g_1) = 1$ we construct a new circuit from C_0 by just adding in the constant gates. We abuse notation and also call this new circuit C_0. The addition of these constant gates to the circuit does not alter the function computed by the circuit, nor does it effect the symmetry of the circuit or whether it has unique labels.

The proof proceeds by first defining a circuit C_1 from C_0 and then defining C' from C_1. We then show that each of these constructions preserves the relevant circuit properties and that C' is reduced and injective.

We now define the circuit C_1. Let $C_1 := \langle G_1, \Omega_1, \Sigma_1, \Lambda_1, L_1\rangle$ be defined as follows. Let $G_1 = G_0 \uplus \{g_\vee\}$. Let $\Omega_1 = \Omega_0$ and $\Lambda_1 = \Lambda_0$. Let $g \in G_1$. If $g = g_\vee$ then $\Sigma_1(g) = \mathrm{OR}[2]$ with $L_1(g)(1) = g_0$ and $L_1(g)(2) = g_1$. If $g \in G_0$, $\Sigma(g) = \mathrm{OR}[2]$ and $H_g = \{g_0, g_1\}$, then $\Sigma_1(g) = \mathrm{OR}[3]$, $L_1(g)(1) = L_0(g)(1)$, $L_1(g)(2) = L_0(g)(2)$, and $L_1(g)(3) = g_\vee$. If $g \in G_0$ and $g = \mathrm{AND}[k]$ for some $k \in \mathbb{N}$ then $\Sigma_1(g) = \mathrm{AND}[k+1]$ and $L_1(g)(i) = L_0(g)(i)$ for all $i \in [k]$ and $L_1(g)(k+1) = g_\vee$. Otherwise let $\Sigma_1(g) = \Sigma_0(g)$ and $L_1(g) = L_0(g)$.

Stated more informally, we define C_1 from C_0 by adding in an OR-gate $g_\vee$ that always evaluates to one, and then adding a wire from that gate to all AND-gates in the circuit (and also a wire from $g_\vee$ to any two-input OR-gate that may already exist in the circuit in order to ensure that $g_\vee$ is part of a singleton syntactic-equivalence class in C_1). The important point to note is that each AND-gate in C_1 has fan-in at least two. We construct C' from C_1 by adding in a number of AND-gates with fan-in one. Therefore, since all of the AND-gates in C_1 have fan-in two, it follows that none of these new gates are syntactically-equivalent to any gate in C_1. We now show that C_1 and C_0 compute the same function and if C_0 is symmetric then C_1 is symmetric.

It is easy to see that if C has order n then C_0 has order n and so C_1 has order n. Let $\mathcal{A}$ be a ρ-structure of size n and let γ be a bijection from the universe of $\mathcal{A}$ to $[n]$. We have that $C_1[\gamma\mathcal{A}](g_\vee) = 1$. We constructed C_1 from C_0 by adding a single wire from $g_\vee$ to each AND-gate and each two-input OR-gate with only the two constant gates as children. Notice that if g is a two-input OR gate in C_0 with the two constant gates as children, then

since g has g_1 as a child and g is an OR-gate $C_0[\gamma\mathcal{A}](g) = C_1[\gamma\mathcal{A}](g) = 1$. It can be shown by induction that if $g \in G_1 \setminus \{g_\vee\}$ then $C_0[\gamma\mathcal{A}](g) = C_1[\gamma\mathcal{A}](g)$. Since $\Omega_1 = \Omega_0$, it follows that C_0 and C_1 compute the same function. We thus have from Lemma 7.3 that C and C_1 compute the same function.

Suppose C is symmetric. From Lemma 7.3 it follows that C_0 is symmetric. Let $\sigma \in \mathbf{Sym}_n$ and let $\pi_0 \in \mathbf{Aut}(C_0)$ be an extension of σ. Let $\pi_1 : G_1 \to G_1$ such that $\pi_1(g) = \pi_0(g)$ for all $g \in G_0$ and $\pi_1(g_\vee) = g_\vee$. It is easy to see that π_1 is an automorphism of C_1 extending σ. It follows that C_1 is symmetric.

We now show that C_1 is reduced. We have from Lemma 7.3 that C_0 is reduced. Since $g_\vee$ is the only two input OR-gate with exactly the two constant gates as children, $g_\vee$ is contained in a singleton syntactic-equivalence class. It can be shown by induction that if $g, g' \in G_0$ are syntactically equivalent in C_1 then they must be syntactically-equivalent in C_0. It follows from these two observations that if $g, g' \in G_1$ are syntactically-equivalent in C_1 then $g = g'$. We thus have that each syntactic-equivalence class in C_1 is a singleton, and so C_1 is reduced.

Let $C' := \{G', \Omega', \Sigma'\Lambda'L'\}$ be defined as follows. For each $g, h \in G_1$ let $c_g^h := |L^{-1}(\{h\})|$. For each $h \in G_1$ let $c^h = \max_{g \in G_1} c_g^h$. For each $h \in G_1$ if $c^h > 1$ we define for each $i \in [c_h - 1]$ a distinct gate g_i^h and let $G_h := \{g_1^h, \ldots, g_{c_h-1}^h\}$, and otherwise let $G_h := \emptyset$. Let $G_\wedge := \biguplus_{h \in G_1} G_h$ and $G' = G_\wedge \uplus G_1$. Let $\Omega' = \Omega_1$ and $\Lambda' = \Lambda_1$. For $g \in G'$ if $g \in G_1$ let $\Sigma'(g) = \Sigma_1(g)$ and otherwise let $\Sigma'(g) = \text{AND}[1]$. For each $g \in G_1$ and $h \in H_g$ let $x_0^{h,g}, \ldots, x_{c_h^g-1}^{h,g}$ be a (0-based) ordering of $L_1(g)^{-1}(\{h\})$. For each $g \in G'$ and $x \in \text{ind}(g)$ we define $L'(g)(x)$ as follows. If $g \in G_1$ then there exists unique $h \in H_g$ and $i \in \{0, \ldots, c_g^h - 1\}$ such that $x = x_i^{h,g}$, and we let $L'(g)(x) = h$ if $i = 0$ and $L'(g)(x) = g_i^h$ otherwise. If $g \in G_\wedge$ then $g = g_i^h$ for some $h \in G_1$ and $i \in [c^h - 1]$, and we let $L'(g)(x) = h$ if $i = 1$ and $L'(g)(x) = g_{i-1}^h$ otherwise.

The construction ensures that C' has injective labels. Let $g \in G_1$. In order to avoid confusion we let H_g be the set of children of g in C_1 and H'_g be the set of children of g in C'. If $g' \in G_1$ we let $g \equiv g'$ denote syntactic-equivalence in C_1 and $g \equiv' g'$ denote syntactic-equivalence in C'.

We now prove that C' has all of the requisite properties.

Claim 7.4.1. *C' is reduced*

Proof. We prove this result by induction on depth. Let $g \in G'$. If g has depth 0 then g is an input gate and so $[g] = \{g\}$. Let $g \in G'$ be an internal gate in C' and suppose for each gate h of depth less than g we have that $[h] = \{h\}$. Let $g' \in G'$ and suppose $g \equiv' g'$. We now show that $g = g'$, and so $[g] = \{g\}$, breaking down the argument by case. We first make a few useful observations. Note that, since $g \equiv' g'$, it follows that $\Sigma(g) = \Sigma(g')$ and both g and g' have the same depth. Moreover, from the inductive hypothesis and the fact that $g \equiv' g'$, we have that there exists $\lambda \in \mathbf{Aut}(g)$ such that for all $x \in \text{ind}(g)$, $L'(g)(\lambda x) = L'(g')(x)$. It follows that $H'_g = H'_{g'}$. We also have from the inductive hypothesis that each child of g and

g' must be contained in a singleton equivalence class. Since C' has injective labels it follows that g and g' have unique labels.

Suppose $g \in G_\wedge$. Then there exists $h \in G_1$ and $i \in [c^h - 1]$ such that $g = g_i^h$. Suppose $g' \in G_1$. Then $\Sigma'(g) = \text{AND}[1]$. But, from the construction of C_1, there are no single-input AND-gates in G_1. It follows $\Sigma'(g') \neq \Sigma'(g)$ and so $g' \not\equiv' g$, a contradiction, and so we must have $g' \in G_\wedge$. If $i = 1$ then $\{h\} = H'_g = H'_{g'}$. Since the only gate in $G_\wedge$ that has h as a child is g_1^h, it follows that $g = g_1^h = g'$. If $i > 1$ then $\{g_{i-1}^h\} = H'_g = H'_{g'}$. Since g_i^h is the only gate in $G_\wedge$ that has g_{i-1}^h as a child, we have $g = g_i^h = g'$. It follows that if $g \in G_\wedge$ then $g = g'$.

Suppose $g \in G_1$. We have already shown that if $g' \in G_\wedge$ then $g \not\equiv' g'$, a contradiction, and so we must have $g' \in G_1$. Since $g \equiv' g'$ there exists $\lambda \in \mathbf{Aut}(g)$ such that for all $x \in \text{ind}(g)$, $L'(g)(\lambda x) = L'(g')(x)$. From the construction, we have that for all $x \in \text{ind}(g)$ and $h \in G_1$, $L'(g)(x) \in \{h, g_1^h, \ldots, g_{c_g^h-1}^h\}$ if, and only if, $L_1(g)(x) = h$. Suppose g is not a trivially invariant gate. Since C is transparent, C_0 is transparent and so C_1 is transparent. We thus have that g and g' have unique labels in C_1 and so $c_g^h = 1 = c_{g'}^h$ and so for all $x \in \text{ind}(g)$, $L_1(g)(x) = L'(g)(x)$ and $L_1(g')(x) = L'(g)(x)$. It follows that for all $x \in \text{ind}(g)$, $L_1(g')(x) = L'(g')(x) = L'(g)(\lambda x) = L_1(g)(\lambda x)$. We thus have that $g \equiv g'$ and so $g = g'$. Suppose instead that g is a trivially invariant gate. Let $x \in \text{ind}(g)$ and $h := L_1(g)(\lambda x)$. Then $L'(g')(x) = L'(g)(\lambda x) \in \{h, g_1^h, \ldots, g_{c_g^h-1}^h\}$. It follows that $L_1(g')(x) = h$. Putting this together we have that for all $x \in \text{ind}(x)$, $L_1(g)(\lambda x) = L_1(g')(x)$, and so $g \equiv g'$ and thus $g = g'$. This completes the proof of Claim 7.4.1. □

We have already shown that if C is symmetric then C_1 is symmetric. We now show that if C_1 is symmetric then C' is symmetric. Suppose C_1 is symmetric. Let $\sigma \in \mathbf{Sym}_n$ and let $\pi_1 \in \mathbf{Aut}(C_1)$ be an extension of σ. We define the function $\pi' : G' \to G'$ as follows. Let $g \in G'$. If $g \in G_1$ let $\pi'(g) := \pi_1(g)$. If $g \notin G_1$ then $g \in G_\wedge$, and so there exists $h \in G_1$ and $i \in [c^h - 1]$ such that $g = g_i^h$. Let $\pi'(g) := g_i^{\pi_1 h}$. It is easy to show that π' is an automorphism of C' extending σ.

Claim 7.4.2. *Let $g \in G_1$, $\mathcal{A}$ be a ρ-structure of size n, and γ be a bijection from the universe of $\mathcal{A}$ to $[n]$. Then $C'[\gamma\mathcal{A}](g) = C_1[\gamma\mathcal{A}](g)$.*

Proof. It is easy to see that for all $h \in G_1$ and $i \in [c^h - 1]$ we have that $C'[\gamma\mathcal{A}](g_i^h) = C'[\gamma\mathcal{A}](h)$. Suppose $g \in G_1$. We now prove the result by induction on the depth of a gate. Suppose g has depth 0. In this case g is an input gate, and the result follows trivially. Suppose g is an internal gate, and for all h of depth less than g we have that if $h \in G_1$ then $C'[\gamma\mathcal{A}](h) = C_1[\gamma\mathcal{A}](h)$. Let $x \in \text{ind}(g)$. Recall that C' is constructed such that if $g \in G_1$ then for all $x \in \text{ind}(g)$ and $h \in H_g$, $L_1(g)(x) = h$ if, and only if, $L'(g)(x) \in \{h, g_1^h, \ldots, g_{c_g^h-1}^h\}$. But, from the construction, we have that all of the gates in $\{h, g_1^h, \ldots, g_{c_g^h-1}^h\}$ evaluate to the same value for a given input to the circuit. Thus, from the inductive hypothesis, we have that if $L'(g)(x) \in G_1$ then $L'^{\gamma\mathcal{A}}(g)(x) = C'[\gamma\mathcal{A}](L(g)(x)) = C_1[\gamma\mathcal{A}](L_1(g)(x)) = L_1^{\gamma\mathcal{A}}(g)(x)$.

If $L'(g)(x) \notin G_1$ then $L'(g)(x) \in G_\wedge$ and so $L'(g)(x) = g_i^h$, where $h = L_1(g)(x)$ and some $i \in [c_g^h - 1]$. But then $L'^{\gamma\mathcal{A}}(g)(x) = C'[\gamma\mathcal{A}](g_i^h) = C'[\gamma\mathcal{A}](h) = C_1[\gamma\mathcal{A}](h) = L_1^{\gamma\mathcal{A}}(g)(x)$. The penultimate equality follow from the inductive hypothesis. The final equality follows from the fact that $h = L_1(g)(x)$. We thus have $C'[\gamma\mathcal{A}](g) = \Sigma'(L'^{\gamma\mathcal{A}}(g)) = \Sigma_1(L_1^{\gamma\mathcal{A}}(g)) = C_1[\gamma\mathcal{A}](g)$. This completes the proof of Claim 7.4.2 □

Since $\Omega' = \Omega_1$, we have that C' and C_1 compute the same function. It follows that, since C_1 and C compute the same function, C and C' compute the same function. Since C is transparent, we may construct the quotient circuit C_0 in time polynomial in $|C|$. Since C_1 is constructed by adding in a single gate and then adding at most $|C_0|$ wires, we may construct C_1 from C_0 in time polynomial in $|C|$. It is easy to see that C' can be constructed in time polynomial in $|C_1|$ and hence polynomial in $|C|$. This completes the proof of the lemma. □

We now show that there is an algorithm that runs in polynomial time and takes as input a circuit with unique labels and an appropriate permutation and outputs the action of the automorphism extending the permutation (if it is defined) on the gates of the circuit.

Lemma 7.5. *There is an algorithm that takes as input a $(\mathbb{B}, \rho)$-circuit C of order n with unique labels and $\sigma \in \boldsymbol{Sym}_n$ and outputs for each gate g the image of g under the action of the unique automorphism extending σ (if it exists). This algorithm runs in time polynomial in the combined size of the input circuit and the encoding of the permutation.*

Proof. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$. Let $C_\equiv = \langle G_\equiv, \Omega_\equiv, \Sigma_\equiv, \Lambda_\equiv, L_\equiv\rangle$ be a quotient of C. We recursively build up the mapping $\pi' \in \mathbf{Aut}(C_\equiv)$ extending σ. If at some point in the recursive construction we arrive at a point where no mapping for g can be found we halt at that point and return that no automorphism exists.

Let h be any gate in $C_\equiv$. Suppose h is an input gate. If h is a constant gate then let $\pi'(h) = h$. If h is a relational gate such that $R := \Sigma_\equiv(h)$, then check if there exists h' such that $\Sigma_\equiv(h') = R$ and $\sigma\Lambda_\equiv(h) = \Lambda_\equiv(h')$, and also check that either both h and h' are output gates or neither are output gates. If no such h' exists then halt and output that no automorphism exists. If neither h nor h' are output gates then set $\pi'(h) = h'$. If both h and h' are output gates then check if $\sigma\Omega_\equiv^{-1}(h) = \Omega_\equiv^{-1}(h')$. If the equality holds set $\pi'(h) = h'$, otherwise halt and output that no automorphism exists. We note that, from the definition of an automorphism, there is at most one such h' meeting this criteria, and so $\pi'(h)$ is well-defined.

Let h be an internal gate in the circuit and assume we have defined $\pi'(g)$ for every gate g of depth less than h. Let h' be a gate in the circuit such that $\Sigma_\equiv(h) = \Sigma_\equiv(h')$, $\pi' L(h)$ is isomorphic to $L_\equiv(h')$ and, if h is an output gate then h' is an output gate such that $\sigma\Omega_\equiv^{-1}(h) = \Omega_\equiv^{-1}(h')$. Suppose there is another gate h'' in the circuit that meets all of those criteria as well. But then it follows that $h'' \equiv h'$. Since $C_\equiv$ is reduced, we then have that $h'' = h'$. We thus have that the choice of h', if it exists, is unique. Note that, since C has

unique labels $C_{\equiv}$ has unique labels, and so unique extensions, and so we have that $\pi' L_{\equiv}(h)$ is isomorphic to $L_{\equiv}(h')$ if, and only if, $L_{\equiv}(h')^{-1}\pi' L_{\equiv}(h)$ acts on $\mathrm{ind}(h)$ like an automorphism of $\mathrm{str}(h)$. This is easy to determine. If no such h' exists, halt and output that there is no automorphism extending σ. Otherwise, let $\pi'(h) = h'$.

It is easy to see that if there is an automorphism of $C_{\equiv}$ extending σ then this construction must have been successful and π' is the unique automorphism of $C_{\equiv}$ extending σ. Thus, if we have thus far halted and returned that no automorphism exists, then indeed there is no automorphism of $C_{\equiv}$ extending σ and so, from Lemma 7.3, no automorphism of C extending σ. We suppose the algorithm has not halted, and thus that π' is an automorphism of $C_{\equiv}$ extending σ.

We say that a function $\pi : G \to G$ is a *pseudo-automorphism* extending σ if (i) π acts like an automorphism extending σ on the input and output gates, (ii) $\Sigma(\pi(h)) = \Sigma(h)$ for all $h \in G$, (iii) $\pi(h) \in H_{\pi(g)}$ for all $h \in G$ and $g \in W(h, \cdot)$, and (iv) $\pi(h) \in \pi'([h])$.

Claim 7.5.1. *If π is an automorphism of C extending σ then π is a pseudo-automorphism extending σ.*

Proof. Suppose π is an automorphism of C extending σ. It is easy to see that conditions (i), (ii) and (iii) are satisfied. We now show that (iv) is satisfied as well. Notice that, since $C_{\equiv}$ is reduced and so has unique extensions, π' is the unique automorphism of $C_{\equiv}$ extending σ. We have from Lemma 7.3 that there exists $\pi_{\equiv} \in \mathbf{Aut}(C_{\equiv})$ extending σ such that $\pi_{\equiv}([g]) = [\pi(g)]$. It follows that for all $h \in G$, $\pi(h) \in \pi_{\equiv}([h]) = \pi'([h])$. □

We now describe an algorithm that takes as input a circuit C and a permutation σ and if there is no pseudo-automorphism extending σ halts and outputs a corresponding message or otherwise halts outputs a pseudo-automorphism π extending σ. This algorithm works by first defining π on the set of output gates and then, by backwards induction on the maximal length of a path from a gate to an output gate, extending the definition of π to the rest of the circuit. Importantly, we have that π, if it exists, is the *unique* pseudo-automorphism extending σ.

For each gate h in C let $Q(h)$ be the maximum length of a path from h to an output gate. Let h be a gate in C. Suppose $Q(h) = 0$. Then h is an output gate. Let $h' \in G$ be such that $h' = \Omega(\sigma\Omega^{-1}(h))$ and $\Sigma(h) = \Sigma(h')$. If there is no such h' then halt and output that no extension exists. Otherwise let $\pi(h) = h'$.

Suppose h is a gate in the circuit with $Q(h) > 0$ and for all g such that $Q(g) < Q(h)$ we have defined $\pi(g)$. Let $H = \bigcap_{g \in W(h,\cdot)} H_{\pi(g)}$. If H is empty then we cannot satisfy condition (iii) in the definition of a pseudo-automorphism, and so we halt and output that no extension exists. Suppose that H is non-empty. If h is an input gate then there is an obvious action of σ on h, and we let $h' = \sigma h$. If h is an output gate, let $h' \in G$ be such that $h' = \Omega(\sigma\Omega^{-1}(h))$ and $\Sigma(h) = \Sigma(h')$. If $h' \notin H$ we halt and output that no extension exists, and otherwise we

let $\pi(h) = h'$. Notice that in the case that h is an output gate or an input gate then from the definition of an automorphism h' is the unique gate satisfying these criteria. Suppose h is an internal non-output gate. Let $h_{\equiv} \in G_{\equiv}$ be such that $h_{\equiv} = [h]$. Let $h' \in \pi_{\equiv}(h_{\equiv})$ be such that $h' \in H$. We now show that h', if it exists, is the unique gate satisfying these criteria. Let $h'' \in \pi_{\equiv}(h_{\equiv})$ and $h'' \in H$. Then $h'' \equiv h'$ and for all $g \in W(h, \cdot)$, $h' \in H_g$ and $h'' \in H_g$. But then, since every $g \in W(h, \cdot)$ has unique labels, $h'' = h'$. Let $\pi(h) = h'$.

It is easy to see that if there is a pseudo-automorphism of C extending σ then this construction must have been successful, and π is the unique pseudo-automorphism extending σ. It follows that if the algorithm has halted without outputting a pseudo-automorphism, then there is no pseudo-automorphism extending σ and so, from the claim, there is no automorphism extending σ. We suppose then that the algorithm has not halted and we have constructed π successfully. It follows from the claim and the uniqueness of π that if there is an automorphism extending σ than it must be equal to π, and so π is an automorphism. We thus have that there is an automorphism extending σ if, and only if, π is an automorphism. It remains to check that π is an automorphism. It suffices to check that π is a bijection and that for each $h \in G$, $\pi L(h)$ is isomorphic to $L(h')$. It is easy to check that π is a bijection. Notice that $\pi L(h)$ is isomorphic to $L(\pi(h))$ if, and only if, $L(\pi(h))^{-1}\pi L(h)$ is an automorphism. This condition is also easy to check. If either of these checks fail, halt and output that there is no automorphism extending σ. Otherwise, output $\pi(g)$ for all $g \in G$.

We note that, since C has unique labels, we can compute $C_{\equiv}$ in polynomial-time. Moreover, it is easy to see that the construction of π' and π can be completed in polynomial-time. We thus have that the algorithm described may be implemented so as to run in polynomial-time. □

We now use Lemma 7.5 to define an algorithm that computes in polynomial-time the image of a given element of the universe of a gate under the action of a given permutation.

Lemma 7.6. *There is an algorithm that takes as input a $(\mathbb{B}, \rho)$-circuit C with unique labels of order n, a gate $\sigma \in \boldsymbol{Sym}_n$, g a gate in C, and $a \in unv(g)$ and, if there exists an automorphism of C extending σ such that $\sigma \in \boldsymbol{Stab}(g)$, outputs $\sigma(a)$. The algorithm runs in time polynomial in the size of C and the encoding of σ.*

Proof. Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$. We use the algorithm from Lemma 7.5 to check if σ extends to an automorphism on C. We also check if $\sigma \in \mathbf{Stab}(g)$. If either of these checks fail, halt and return that no such automorphism exists. Let $h \in H_g$ and $\vec{b}_R := L(g)^{-1}(h)$ be such that $a \in \vec{b}_R$, and let i be the index of a in $\vec{b}_R$. Halt and output $\sigma a = (L(g)^{-1}(\sigma h))(i)$. □

We aim to show that it is possible to compute in polynomial-time the orbits and canonical supporting partitions of the gates, and elements of the universes of the gates, of a given circuit with unique labels. In order to prove this, we first prove a more general result which shows that there is a polynomial-time algorithm that takes as input a set X, an element

$x \in X$, and a polynomial-time computable group action on X, and computes the orbit and canonical supporting partition of x.

Lemma 7.7. *Let p be a polynomial. There is an algorithm that takes as input a set $S \subseteq [n]$, a set X, an element $x \in X$, and a Turing machine T computing the action of $\boldsymbol{Stab}(S)$ on X that runs in time bounded by $p(n + |X|)$, and outputs $\boldsymbol{Orb}_{\boldsymbol{Stab}(S)}(x)$ and $\boldsymbol{SP}_{\boldsymbol{Stab}(S)}(x)$. This algorithm runs in time polynomial in $n + |X| + |T|$.*

Proof. Let $(u, v) \in \mathbf{Sym}_{[n]\setminus S}$ be a transposition. We note that $(u, v) \in \mathbf{Stab}(S)$ and there are $\binom{n-|S|}{2}$ many such transpositions. For each $(u, v) \in \mathbf{Sym}_{[n]\setminus S}$ let

$$\mathcal{P}_{(u,v)} := \{\{u, v\}\} \cup \bigcup_{w \in [n]\setminus\{u,v\}} \{\{w\}\}.$$

We note that $\mathcal{P}_{(u,v)}$ is a partition of $[n]$. Then $\mathcal{P}_{(u,v)}$ supports $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$ if, and only if, $(u, v) \cdot x = x$.

Let $\mathcal{P}$ be the partition that is the coarsest common refinement of the partitions $\mathcal{P}_{(u,v)}$, for all u, v with $(u, v) \cdot x = x$. From Proposition 6.3 we have that $\mathcal{P}$ supports $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$. Suppose that $\mathcal{P}$ is not the coarsest supporting partition of $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$. Then there exists a partition $\mathcal{P}'$ supporting $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$ such that $\mathcal{P}' \preceq \mathcal{P}$ and $\mathcal{P}' \neq \mathcal{P}$. And so there exists $P \in \mathcal{P}$ and $P' \in \mathcal{P}'$ such that $P \subsetneq P'$. But then there exists $a, b \in P'$ such that $a \notin P$ and $b \in P$. Note that (a, b) fixes $\mathcal{P}'$ and, since $\mathcal{P}'$ supports $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$, it follows that $(a, b) \in \mathbf{Stab}_{\mathbf{Stab}(S)}(x)$ and $a \notin S$ and $b \notin S$. But then we have that $(a, b) \cdot x = x$, and so $\mathcal{P}_{(a,b)}$ supports $\mathbf{Stab}_{\mathbf{Stab}(S))}(x)$ and thus, from the construction of $\mathcal{P}$, $\mathcal{P}$ is fixed by (a, b). But we selected a and b such that $\mathcal{P}$ is not fixed by (a, b), and so we have a contradiction. We thus have that $\mathcal{P}$ is the coarsest supporting partition of $\mathbf{Stab}_{\mathbf{Stab}(S)}(x)$.

It remains to compute $\mathbf{Orb}_{\mathbf{Stab}(S)}(x)$. Let $M_0 := \{x\}$ and for each $i \geq 0$ let $M_{i+1} := M_i \cup (\bigcup_{(u,v)\in\mathbf{Sym}_{[n]\setminus S}}((u, v) \cdot M_i))$. Let $M \subseteq X$ be the union of this sequence. It is easy to see that $M \subseteq \mathbf{Orb}_{\mathbf{Stab}(S)}(x)$ as every element of M is equal to the action of some finite sequence of transpositions acting on x. Moreover, if $y \in \mathbf{Orb}_{\mathbf{Stab}(S)}(x)$, then there exists $\pi \in \mathbf{Stab}(S)$ such that $y = \pi \cdot x$. But then, since $\mathbf{Sym}_{[n]\setminus S}$, is generated by the set of all transpositions in $\mathbf{Sym}_{[n]\setminus S}$, it follows that π can be written as a sequence of t transpositions for some $t \in \mathbb{N}$. Thus $y \in M_t \subseteq M$, and hence $\mathbf{Orb}_{\mathbf{Stab}(S)}(x) \subseteq M$, and so $\mathbf{Orb}_{\mathbf{Stab}(S)}(x) = M$.

Note that the set of all transpositions in $\mathbf{Sym}_{[n]\setminus S}$ can be computed in time $\mathcal{O}(n^2)$, and we can check if a given transposition fixes x by simulating T with the given transposition and x as inputs. Moreover, since it is easy to show that $\mathcal{E}$ (the operator defined in Section 6.1) can be computed in time polynomial in n, it follows that $\mathcal{P}$ can be computed in $\mathcal{O}(n^2|T|p(|X| + n)^2 q(|X| + n))$, for some polynomial q. The dependence of the running time on the size of T is a result of the overhead required to simulate T.

Furthermore, when computing the orbit, we construct M_i iteratively and obtain the orbit after at most $|X|$ iterations. Since each iteration requires at most $\mathcal{O}(n^2)$ applications of the

group action, it follows that this part of the procedure runs time $\mathcal{O}(n^2|X||T|p(|X|+n)^2)$. We thus have that the entire algorithm runs in polynomial-time, and the result follows. □

We now apply Lemma 7.7 and show that there is a polynomial-time algorithm that takes as input a circuit with unique labels and decides if the circuit is symmetric and, if it is, outputs the orbit and canonical supporting partition of each gate in the circuit.

Lemma 7.8. *There is an algorithm that takes in a circuit C with unique labels and outputs if the circuit is symmetric. If it is symmetric then it outputs the orbit and coarsest supporting partition of each gate. This algorithm runs in time polynomial in the size of the circuit.*

Proof. Let n be the order of C. We have from Lemma 7.5 that there is a Turing machine T' that takes as input a circuit with unique labels and a permutation and outputs the image of each gate (if it exists) in polynomial-time. We define a Turing machine T that takes as input a permutation $\sigma \in \mathbf{Sym}_n$ and a gate g in C, runs T' with inputs C and σ and outputs the image of g under the action of σ (if it exists).

For each transposition $(u, v) \in \mathbf{Sym}_n$ and each gate g in C we use T to check if the image of g under the action of (u, v) exists. If for any transposition and gate T returns that no image exists then we halt and output that the circuit is not symmetric.

We note that if every gate has an image under the action of every transposition then, since $\mathbf{Sym}_n$ is generated by the set of transpositions, we have that C is symmetric.

For each gate g in C we run the algorithm from Lemma 7.7 with $S := \emptyset$, $X := G$ (where G is the set of gates in C), $x := g$, and Turing machine T, and output the result of this computation.

We note that there are $\binom{n}{2} \leq n^2$ transpositions in $\mathbf{Sym}_n$ and so, since from Lemma 7.5 the action of a transposition on the gates of the circuit can be computed in polynomial-time, the initial symmetry check can be completed in polynomial-time. Moreover, from the polynomial-time bounds in Lemmas 7.5 and 7.7, we have that the rest of the algorithm also runs in time polynomial in the size of the circuit. □

We now extend Lemma 7.8 and construct a polynomial-time algorithm that computes the orbit and canonical supporting partition of each element of the universe of each gate in a circuit.

Lemma 7.9. *There is an algorithm that takes in a circuit C of order n with unique labels, a gate g in C with small support, and $a \in unv(g)$, and outputs if the circuit is symmetric. If C is symmetric it outputs the orbit $\boldsymbol{Orb}_{sp(g)}(a)$ and coarsest supporting partition $\boldsymbol{SP}_{sp(g)}(a)$. This algorithm runs in time polynomial in the size of the circuit.*

Proof. We first use the algorithm from Lemma 7.8 to compute the canonical support of g. If the algorithm returns that C is not symmetric, output that C is not symmetric.

We have from Lemma 7.6 that there is a Turing machine T' that takes as input a circuit with unique labels, a gate, an element of the universe of that gate, and a permutation, and outputs the image of the given element under the action of the given permutation (if it exists). We define a Turing machine T that takes as input an element $b \in \mathrm{sp}(g)$ and a permutation $\sigma \in \mathbf{Stab}(\mathrm{sp}(g))$ outputs the result of running T' with inputs C, σ, g and b.

We then use the algorithm from Lemma 7.7, with inputs $S := \mathrm{sp}(g)$, $X := \mathrm{unv}(g)$ and $x := a$, and the Turing machine T, and output the results.

We have from the bounds in Lemmas 7.6 and 7.7 that this algorithm runs in time polynomial in the size of the circuit. □

We have shown that transparent circuits, and circuits with unique labels, have all of the requisite algorithmic properties needed to establish our main result. However, since transparency is defined in terms of syntactic-equivalence, and testing for syntactic-equivalence reduces to testing for isomorphism, it is not obvious that transparency itself is a polynomial-time decidable property of circuits. If transparency were not polynomial-time decidable then the restriction to transparent circuits in the main theorem of this thesis would be quite unnatural. It may also undermine the usefulness of this result. We now show that the class of transparent circuits is indeed polynomial-time decidable.

Proposition 7.10. *There is an algorithm that takes as input a circuit and decides if that circuit is transparent. This algorithm runs in time polynomial in the size of the circuit.*

Proof. Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit. We now describe an algorithm. We first check that $L(g)$ is an injection for every non-trivially invariant gate $g \in G$. If this is not the case we return that C is not transparent. For each $p \in \mathbb{N}$ let $G^p \subseteq G$ be the set of all gates of depth p and let $G^{\leq p} = \bigcup_{0 \leq i \leq p} G^i$. We have that no two input gates are syntactically-equivalent. It follows that a gate in G^1 has unique labels if, and only if, it has injective labels. We therefore have that all of the non-trivially invariant gates in G^1 have unique labels. We then run the following iterative algorithm. We initialise a variable i to 1. We have that all of the non-trivially invariant gates in $G^{\leq i}$ have unique labels. We can thus compute the syntactic-equivalence classes of $G^{\leq i}$ using the algorithm given in Lemma 7.1. The children of the gates in G^{i+1} are in $G^{\leq i}$. We check if every non-trivially invariant gate in G^{i+1} has unique labels, i.e. if no two of its children are elements of the same syntactic-equivalence class. If this check fails, we halt and output that the circuit is not transparent, and if it succeeds we increment the variable i and continue as above. If i is ever set to the value $\mathrm{depth}(C)$ we halt and output that the circuit is transparent. It is easy to see that this algorithm runs in polynomial-time. □

We can similarly show that the class of circuits with unique labels is polynomial-time decidable.

Corollary 7.11. *There is an algorithm that takes in a circuit and decides if that circuit has unique labels and runs in time polynomial in the size of the circuit.*

Proof. Let C be the input circuit. From Proposition 7.10 we may check if C is transparent in time polynomial in the size of C. If C is not transparent halt and output that C does not have unique labels. If C is transparent then from Lemma 7.1 we may compute the syntactic-equivalence relation for the gates of C in time polynomial in the size of C. Note that C has unique labels if, and only if, for each gate g in C, $|\mathrm{ind}(g)| = |H_g/_{\equiv}|$. We may thus check if C has unique labels by iterating through the gates of C. □

7.2 The Necessity of Transparency

We have shown then that many crucial properties of transparent circuits are polynomial-time decidable and, using these results, we have shown that transparent circuits can be transformed in polynomial-time to equivalent reduced injective circuits. We have also shown that circuits with unique labels have many of the requisite algorithmic properties needed for the proof of our main result.

In this section we formally establish that most of these circuit properties are at least as hard to decide as the graph isomorphism problem. In particular, we present reductions from the graph-isomorphism problem to most of the important decision problems addressed in the first section of this chapter, including: deciding if a circuit is symmetric, deciding if a gate has unique labels, deciding if two gates are syntactically-equivalent, deciding if two gates are in the same orbit, etc. Moreover, we show that many of these hardness results hold even if we restrict ourselves to other natural classes of circuits, such as the class of circuits with injective labels or the class of circuits with unique children.

These results together suggest the necessity of the restriction to transparent circuits. Moreover, while we do not show that there is no polynomial-time computable transformation from a general circuit to an equivalent transparent circuit (or equivalent circuit with unique labels), the difficulty associated with computing these basic circuit properties that seem essential for defining such a transformation should be considered evidence that, at the very least, many of the obvious approaches for defining such an algorithm require showing that graph-isomorphism in P.

In each case we establish this reduction by showing that the graph-isomorphism problem can be encoded as a question about a circuit with non-trivially invariant gates. In each case we use circuits defined over the same basis and it may be asked whether these hardness results might fail to hold if we restrict ourselves to circuits defined over bases from some class that excludes this one. However, it is easy to show that these hardness results can be generalised to a broad range of bases.

We now fix the basis we use in this section. Let $\tau = (\{M\}, \{s_1, s_2\}, \zeta)$ be a many-sorted relational vocabulary such that $\zeta(M) = (s_1, s_2)$. Let $\mathcal{G}$ be a class of τ-structures. Let $\mathbb{B}_{\mathcal{G}}$ be the basis corresponding to $\mathcal{G}$. We fix $\mathbb{B}_{\mathcal{G}}$ for the remainder of this section.

Remark 7.12. In this section we present a number of polynomial-time reductions from the graph-isomorphism problem to various circuit-related problems. In each case we present a reduction from the bipartite-isomorphism problem to a circuit-related problem. This suffices as, from [44], there is a polynomial-time reduction from the graph-isomorphism problem to the bipartite-isomorphism problem. We recall that the bipartite-isomorphism problem is the problem of deciding if for a given pair of bipartite graphs $B_1 := (U_1, V_1, E_1)$ and $B_2 := (U_2, V_2, E_2)$ there exists a (graph) isomorphism $\pi : B_1 \to B_2$ such that $\pi(U_1) = U_2$ and $\pi(V_1) = V_2$.

We now present a reduction from the graph-isomorphism problem to the problem of deciding if two gates in a circuit are syntactically-equivalent. In fact, we prove a stronger result, presenting a reduction from the graph-isomorphism problem to the problem of computing the syntactic-equivalence relation over a more constrained class of circuits.

Proposition 7.13. *There is a polynomial-time reduction from the graph isomorphism problem to the problem of determining if a given pair of gates in a given circuit are syntactically-equivalent.*

Proof. Let ρ be a non-empty relational vocabulary. We reduce the bipartite-isomorphism problem to the problem of deciding whether two given gates are syntactically equivalent in a symmetric $(\mathbb{B}_{\mathcal{G}} \cup \mathbb{B}_{\text{std}}, \rho)$-circuit taking ρ-structures as input where the circuit (i) has injective labels, (ii) contains no constant gates, and (iii) contains at most two non-trivially invariant gates. Suppose we are given two partitioned bipartite graphs $B_1 := (U_1, V_1, E_1)$ and $B_2 := (U_2, V_2, E_2)$. We assume, without a loss of generality, that there exists $a_1, b_1, a_2, b_2 \in \mathbb{N}$ such that $U_1 = [a_1]$, $V_1 = [b_1]$, $U_2 = [a_2]$, and $V_2 = [b_2]$. Let n be the size of these graphs (i.e. $n = a_1 + b_1 = a_2 + b_2$).

The idea is to construct a circuit with n inputs, and with two designated gates used to encode the presence or absence of an edge, and two non-trivially invariant gates wired up so as to encode the two graphs. We wire the circuit such that, for a given non-trivially invariant gate g_{ns}, the child of g_{ns} labelled by (p, q) has exactly one of the two designated gates as a child, with the choice of which one depending on whether (p, q) is an edge in the associated graph or not. In this sense the circuit encodes the two bipartite graphs at the non-trivially invariant gates, and the two non-trivially invariant gates are syntactically-equivalent if, and only if, the two graphs are bipartite-isomorphic. We now present this construction formally.

Let R be a fixed-relation symbol in ρ. Let $G_R := \{g_{R,\vec{c}} : \vec{c} \in [n]^{r_R}\}$, $G_{\text{mid}} := \{g_\vee, g_\wedge, g_{\text{out}}\}$, $G_{\text{ns}} := \{g^1_{\text{ns}}, g^2_{\text{ns}}\}$, and $G_{\text{nodes}} := \{g_{i,(u,v)} : i \in [2], (u, v) \in [a_i] \times [b_i]\}$. Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}_{\mathcal{G}} \cup \mathbb{B}_{\text{std}}, \rho)$-circuit of order n defined as follows. Let $G = G_R \cup G_{\text{mid}} \cup G_{\text{ns}} \cup G_{\text{nodes}}$

and Ω be the 0-ary function g_{out}. For each $\vec{c} \in [n]^{r_R}$ let $\Lambda_R(\vec{c}) = g_{R,\vec{c}}$. Define Σ as follows. For each $g \in G$,

- if $g = g_{\text{out}}$ then $\Sigma(g) = \text{AND}[2]$,
- if $g \in G_{\text{ns}}$ let $i \in [2]$ be such that $g = g^i_{\text{ns}}$ and let $\Sigma(g) = F_{\mathcal{G}}[a_i, b_i]$,
- if $g \in G_{\text{nodes}}$ then $\Sigma(g) = \text{AND}[1]$,
- if $g = g_\wedge$ then $\Sigma(g) = \text{AND}[n^{r_R}]$ and if $g = g_\vee$ then $\Sigma(g) = \text{OR}[n^{r_R}]$, and
- if $g \in G_R$ then $\Sigma(g) = R$.

Define L as follows. For each $g \in G$,

- if $g = g_{\text{out}}$ then for each $i \in [2]$, $L(g)(i) := g^i_{\text{ns}}$,
- if $g \in G_{\text{ns}}$ and $g = g^i_{\text{ns}}$ then for all $(p, q) \in \text{ind}(g)$ let $L(g)(p, q) = g_{i,(p,q)}$,
- if $g \in G_{\text{nodes}}$ and $g = g_{i,(p,q)}$, then if $(p, q) \in E_i$ let $L(g)(1) = g_\wedge$ and otherwise let $L(g)(1) = g_\vee$, and
- if $g = g_\wedge$ or $g = g_\vee$ then for all $q \in [n^{r_R}]$ we have that $L(g)(q) = \Lambda^{-1}(\vec{c}_q)$, where $\vec{c}_q$ is the qth element of $[n]^{r_R}$ in the lexicographical ordering on $[n]^{r_R}$.

We note that for $i \in [2]$ the child of $L(g^i_{\text{ns}})(p, q)$ is $g_\wedge$ if, and only if, (p, q) is an edge in B_i and the child of $L(g^i_{\text{ns}})(p, q)$ is $g_\vee$ if, and only if, (p, q) is not an edge in B_i. We thus have that B_1 and B_2 are bipartite-isomorphic if, and only if, $\Sigma(g^1_{\text{ns}}) = \Sigma(g^2_{\text{ns}})$ and there exists $\lambda \in \text{ind}(g) = [a_1] \times [b_1] = [a_2] \times [b_2]$ such that for all $(u, v) \in [a_1] \times [b_1]$, $L(g^1_{\text{ns}})(u, v) = g_{1,(u,v)} \equiv g_{2,\lambda(u,v)} = L(g^2_{\text{ns}})(\lambda(u, v))$. It follows that B_1 and B_2 are bipartite-isomorphic if, and only if, $g^1_{\text{ns}} \equiv g^2_{\text{ns}}$.

Since the construction of C can be complete in time polynomial in the combined sizes of the input graphs, the mapping of (B_1, B_2) to the tuple $(C, (g^1_{\text{ns}}, g^2_{\text{ns}}))$ is a reduction, and the result follows. □

It follows from the proof of Proposition 7.13 that computing the syntactic-equivalence relation for a given circuit remains hard even if we restrict ourselves to injective circuits.

We now show that the syntactic-equivalence relation remains hard to compute even if we restrict ourselves to the class of circuits such that each non-trivially invariant gate has unique children.

Lemma 7.14. *There is a polynomial-time reduction from the graph-isomorphism problem to the problem of deciding if two gates in a given circuit with the property that each non-trivially invariant gate has unique children are syntactically-equivalent.*

Proof. We use a similar approach as in the proof of Proposition 7.13. The circuit used here is similar except that we omit the children of the non-trivially invariant gates and replace them with direct wires. We now present this reduction formally.

Suppose we are given two partitioned bipartite graphs $B_1 := (U_1, V_1, E_1)$ and $B_2 := (U_2, V_2, E_2)$. We assume, without a loss of generality, that there exists $a_1, b_1, a_2, b_2 \in \mathbb{N}$ such that $U_1 = [a_1]$, $V_1 = [b_1]$, $U_2 = [a_2]$, and $V_2 = [b_2]$.

Let $C = \langle G, , \Omega, \Sigma, \Lambda, L\rangle$ be the circuit defined in the proof of Proposition 7.13. Let $C' = \langle G', \Omega', \Sigma', \Lambda', L'\rangle$ be defined as follows. Let $G' = G \setminus G_{\text{nodes}}$, $\Omega' = \Omega$, $\Lambda' = \Lambda$, and $\Sigma' = \Sigma|_{G'}$. For all $g \in G' \setminus G_{\text{ns}}$ let $L'(g) = L(g)$. For $i \in [2]$ and $(p, q) \in [a_i] \times [b_i]$ let $L(g^i_{\text{ns}})(p, q) = g_\wedge$ if $(p, q) \in E_i$ and $L(g^i_{\text{ns}})(p, q) = g_\vee$ otherwise.

We have defined C' from C by deleting the gates in G_{nodes} and for each $g \in G_{\text{nodes}}$ adding a wire directly from the child of g in to the parent of g. Using an argument similar to that of Proposition 7.13 we have that B_1 and B_2 are bipartite-isomorphic if, and only if, $g^1_{\text{ns}} \equiv g^2_{\text{ns}}$. Since the construction of C, and so C', can be implemented in time polynomial in the combined sizes of the input graphs, the mapping of (B_1, B_2) to the tuple $(C', (g^1_{\text{ns}}, g^2_{\text{ns}}))$ is a reduction, and the result follows. □

We recall that a circuit C is transparent if, and only if, every non-trivially invariant gate in C has injective labels and unique children. We have shown that computing the syntactic-equivalence relation of a circuit is at least as hard as the graph-isomorphism problem even if we restrict ourselves to either the class of circuits in which each non-trivially invariant gate has injective labels (Proposition 7.13) or the class of circuits in which each non-trivially invariant gate has unique children (Lemma 7.14). It follows that while transparency, which is the conjunction of these two properties, suffices to establish the polynomial-time computability of syntactic-equivalence, each of these properties alone is too weak to unconditionally guarantee the existence of a polynomial-time algorithm computing syntactic-equivalence.

We now present a reduction from the problem of deciding the syntactic-equivalence relation to the problem of deciding if a given gate in a circuit has unique labels. It follows from Proposition 7.13 and the composition of reductions that there is a reduction from the graph-isomorphism problem to the problem of deciding if a gate has unique labels.

Lemma 7.15. *There is a polynomial-time reduction from the problem of determining if a given pair of gates in a given circuit are syntactically equivalent to the problem of determining if a given gate in a given circuit has unique labels.*

Proof. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a circuit of order n and let $g_1, g_2 \in G$. Let D be the circuit formed from C by removing every gate $g \in G \setminus \{g_1, g_2\}$ such that $\neg W_t(g, g_1) \wedge \neg W_t(g, g_2)$, where W_t is the transitive closure of the W relation (i.e. we remove all those gates in the circuit such that there is no path from the gate to either g_1 or g_2). Let C' be the circuit formed from D by adding in a single two-input AND-gate g' and connecting the outputs of g_1 and g_2 to the inputs of g'. Moreover, we let this g' be the single output gate of C'.

It follows that g_1 and g_2 are syntactically-equivalent in C if, and only if, g' has unique labels in C'. Since the construction of C' from C can be completed in polynomial time, the mapping of $(C, (g_1, g_2))$ to (C', g') is a reduction. □

We say a gate g in a circuit C has unique extensions if there is no permutation $\sigma \in \mathbf{Sym}_n$ such that there exists two automorphisms extending σ that disagree with each other on g (i.e. g is not a counter example to C having unique extensions). We now use a similar argument as for Lemma 7.15 to establish a reduction from the problem of deciding if two gates are syntactically-equivalent in an injective circuit to the problem of deciding if a given gate in an injective circuit does not have unique extensions. It follows from the composition of reductions and Proposition 7.13 that there is a reduction from the graph-isomorphism problem to the problem of deciding if a gate does not have unique extensions.

Lemma 7.16. *There is a polynomial-time reduction from the problem of determining if a given pair of gates in a given circuit with injective labels are syntactically-equivalent to the problem of determining if for a given pair (C, g), where C is an injective circuit of order n and g is a gate in C, that there exists $\sigma \in \boldsymbol{Sym}_n$ and automorphisms $\pi, \pi' \in \boldsymbol{Aut}(C)$ extending σ such that $\pi(g) \neq \pi'(g)$.*

Proof. Let C be a circuit of order n and let g_1 and g_2 be two gates in C. Note that for any gate g in $\sigma \in \mathbf{Sym}_n$, if $\pi, \pi' \in \mathbf{Aut}(C)$ both extend σ and $\pi_e := \pi'\pi^{-1}$ then $\pi(g) \neq \pi'(g)$ if, and only if, $\pi_e(g) \neq g$. We thus have that there exists $\sigma \in \mathbf{Sym}_n$ and $\pi, \pi' \in \mathbf{Aut}(C)$ extending σ such that $\pi(g) \neq \pi'(g)$ if, and only if, there exists $\pi_e \in \mathbf{Aut}(C)$ extending the trivial permutation such that $\pi_e(g) \neq g$.

Let C' be the circuit constructed from C as in the proof of Lemma 7.15. We now show that the mapping (C, g_1, g_2) to (C', g_1) is a reduction. Let π_e be a bijection from the gates of C' to the gates of C' that swaps g_1 and g_2 and fixes all other gates. It follows that if g_1 and g_2 are syntactically-equivalent in C, then they are syntactically-equivalent in C', and so π_e is a non-trivial automorphism extending the trivial permutation, and thus g_1 does not have unique extensions in C'. We now prove the other direction. Suppose g_1 does not have unique extensions in C'. Then there exists an automorphism $\pi_e \in \mathbf{Aut}(C')$ extending the trivial permutation and such that $\pi_e(g_1) \neq g_1$. But g_1 is a child of the single output gate g' (which must be fixed by any automorphism), and the only other child of g' is g_2. It follows π_e swaps g_1 and g_2, and so from Lemma 4.14 g_1 and g_2 are syntactically-equivalent in C'. The result follows. □

We now show that there is a reduction from the graph-isomorphism problem to the problem of deciding if a circuit is symmetric. In fact, we prove a stronger result, showing that this reduction holds even if we restrict ourselves to the class of reduced circuits in which all but two gates in the circuit have injective labels. We might think of this as the class of circuits that are *almost* reduced and injective.

Proposition 7.17. *The graph-isomorphism problem is polynomial-time reducible to the problem of deciding if a reduced circuit with all but two gates being injective is symmetric.*

Proof. We use an approach similar to that in the proof of Proposition 7.13. In this case we construct a circuit with two inputs, and each input is connected to an approximate copy of the circuit defined in the proof of Lemma 7.14. We now define this reduction formally.

Suppose we are given two partitioned bipartite graphs $B_1 := (U_1, V_1, E_1)$ and $B_2 := (U_2, V_2, E_2)$. We assume, without a loss of generality, that there exists $a_1, b_1, a_2, b_2 \in \mathbb{N}$ such that $U_1 = [a_1]$, $V_1 = [b_1]$, $U_2 = [a_2]$, and $V_2 = [b_2]$.

Let $\rho := \{R\}$ be a relational vocabulary, where R is a unary relational symbol. We define a $(\mathbb{B}_{\mathcal{G}} \cup \mathbb{B}_{\text{std}}, \rho)$-circuit $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ of order two as follows. Let $G_R := \{g_R^1, g_R^2\}$ and $G_{\text{mid}} := \{g_\wedge^1, g_\wedge^2, g_\vee^1, g_\vee^2, g_{\text{out}}\}$, and $G_{\text{ns}} := \{g_{\text{ns}}^1, g_{\text{ns}}^2\}$. Let $G = G_R \cup G_{\text{mid}} \cup G_{\text{ns}}$ and Ω be the 0-ary function g_{out}. Let $\Lambda(1) := g_R^1$ and $\Lambda(2) := g_R^2$. Define Σ as follows. For each $g \in G$,

- if $g = \wedge_{\text{out}}$ let $\Sigma(g) = \text{AND}[2]$,
- if $g \in G_{\text{ns}}$ let $\Sigma(g) = F_{\mathcal{G}}[a, b]$,
- if $g = g_\wedge^1$ or $g = g_\wedge^2$ let $\Sigma(g) = \text{AND}[1]$ and if $g = g_\vee^1$ or $g = g_\vee^2$ let $\Sigma(g) = \text{OR}[1]$, and
- if $g \in G_R$ let $\Sigma(g) = R$.

Define L as follows. For each $g \in G$,

- if $g = g_{\text{out}}$ then for each $i \in [2]$ let $L(g)(i) := g_{\text{ns}}^i$,
- if $g \in G_{\text{ns}}$ and $g = g_{\text{ns}}^i$ for some $i \in [2]$ then for $(p, q) \in [a_i] \times [b_i]$ let $L(g)(p, q) = g_\wedge^i$ if $(p, q) \in E_i$ and $L(g)(p, q) = g_\vee^i$ otherwise, and
- if $g = g_\wedge^i$ or $g = g_\vee^i$ for some $i \in [2]$ let $L(g)(1) = \Lambda^{-1}(i)$.

We note that for $i \in [2]$, $L(g_{\text{ns}}^i)(p, q) = g_\wedge^i$ if, and only if, (p, q) is an edge in B_i and $L(g_{\text{ns}}^i)(p, q)g_\vee^i$ if, and only if, (p, q) is not an edge in B_i. Let $\pi_{(1,2)} : G \to G$ be the function that fixes the output gate, and for each symbol $s \in \{\wedge, \vee\}$ swaps the gate g_s^1 with the gate g_s^2. It is easy to see that $\pi_{(1,2)}$ is a bijection. Moreover, C is symmetric if, and only if, $\pi_{(1,2)}$ is an automorphism of the circuit extending the transposition $(1, 2)$, if and only if, $(1, 2)(L(g_{\text{ns}}^1))$ is isomorphic to $L(g_{\text{ns}}^2)$ if, and only if, B_1 and B_2 are bipartite-isomorphic.

We also note that every gate in the circuit is part of a singleton syntactic-equivalence class and all of the gates in the circuit except for the two non-trivially invariant gates have unique labels. Since the construction of C can be implemented in time polynomial in the combined sizes of the input graphs, the mapping of (B_1, B_2) to C is a reduction, and the result follows. □

It follows from Proposition 7.17 that deciding if a circuit is symmetric is at least as hard as the graph-isomorphism problem even if we restrict ourselves to reduced circuits or circuits with unique children. Moreover, it is possible to alter this reduction, using a construction analogous to the one used in the proof of Proposition 7.13, and show that the problem of deciding symmetry for injective circuits is also as hard as the graph-isomorphism problem. In contrast, we have from Lemma 7.9 that we can decide if a transparent circuit is symmetric

in polynomial-time. These observations again suggest both the robustness of this hardness result and the importance of the transparency condition.

In Lemma 7.5 we showed that for circuits with unique labels we can compute the action of an automorphism on the gates of a circuit in polynomial-time. In Proposition 7.8 we show that we can also compute the orbit and supports of gates in polynomial-time. These results play a central role in our translation from families of circuits to formulas, and hence in the proof of our main result. We now show that deciding the orbit of a gate in a general circuit is at least as hard as the graph isomorphism problem. We show that this result holds even if we restrict our attention to circuits with unique extensions or circuits with injective labels.

Lemma 7.18. *There is a polynomial-time reductions from the graph-isomorphism problem to the problem of deciding if two given gates in a given symmetric circuit are in the same orbit. This result holds even if we restrict ourselves either to circuits with unique extensions, to circuits with unique children, or to circuits with injective labels.*

Proof. Let B_1 and B_2 be bipartite graphs. Let C be the circuit constructed in Lemma 7.14. The mapping of (B_1, B_2) to $(C, g^1_{\mathrm{ns}}, g^2_{\mathrm{ns}})$ is a reduction from the graph isomorphism problem to the problem of deciding if two gates in a circuit with unique children (and so unique extensions) are in the same orbit. A similar reduction using the circuit constructed in Proposition 7.13 gives a reduction to the problem of deciding if two gates in a given circuit with injective labels are in the same orbit. □

We have from Propositions 7.13 and 7.17 and from Lemmas 7.15, 7.14, 7.18, and 7.16, that a number of basic circuit properties are at least as hard for general circuits a the graph isomorphism problem and that this hardness results hold even if we restrict our attention to particular classes of circuits. In contrast, we proved in Section 7.1 that all of these properties are known to be polynomial-time decidable for transparent circuits.

It is worth noting that we have not established the hardness of computing a function that maps a circuit to an equivalent transparent circuit. However, we have shown that many of the related decision problems are at least as hard as the graph-isomorphism problem. This excludes many of the obvious translations from circuits to transparent circuits that require computing, for example, the syntactic-equivalence relation. In particular, the translation from circuits to reduced circuits presented by Anderson and Dawar [3] explicitly uses the polynomial-time computability of syntactic-equivalence for circuits with trivially invariant gates, and does not generalise to our setting. As such, while we do not establish the hardness of computing this function we consider the results presented in this section to be evidence against the existence of an easily-definable polynomial-time translation from general circuits to transparent circuits, or to circuits with unique labels, assuming the graph isomorphism problem is not in polynomial-time.

We now establish one final hardness result. We have from Proposition 7.10 and Corollary 7.11 that the set of transparent circuits and the set of circuits with unique labels are

both polynomial-time decidable. However, we now show that other important classes of circuits, such as the class of circuits with unique extensions, are as hard to decide as the graph-isomorphism problem.

Lemma 7.19. *There are polynomial-time reductions from the graph isomorphism problem to the problem of deciding if a circuit does not have unique extensions and the problem of deciding if a circuit does not have unique children.*

Proof. In both cases the circuit constructed in the proof of Lemma 7.14 suffices for the reduction. □

Chapter 8

Translating Circuits to Formulas

In Chapter 5 we showed that for any set of almost relational P-bounded generalised operators $\mathbf{\Omega}$ every query definable in $\mathrm{FP}(\widetilde{\mathbf{\Omega}})$ is definable by a P-uniform family of transparent symmetric circuits defined over the basis $\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathbf{std}}$. In this chapter we prove a partial converse to this result and establish a translation from P-uniform families of transparent symmetric circuits to formulas in the corresponding extension of fixed-point logic. The formal statement of this result is as follows.

Proposition 8.1. *Let $\mathbf{\Omega}$ be a finite set of almost relational* P*-bounded vectorised operators. Let ρ be a relational vocabulary. Every query definable by a* P*-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{std}, \rho)$-circuits $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ is definable in $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})[\rho]$.*

The proof of this proposition is structured as follows. First, we use the Immerman-Vardi theorem to show that the P-uniform family $(C_n)_{n\in\mathbb{N}}$ is definable by an $\mathrm{FP}^{\mathbb{N}}$-interpretation in the number sort (thought of as a linearly ordered structure). We use Lemma 7.4 to show that each C_n can be assumed to have unique labels and use the support theorem to show that each gate and element of the universe of a gate has constant size support. We show that the evaluation of a gate g in C_n for an input structure $\mathcal{A}$ and bijection $\gamma \in [n]^{\underline{A}}$ is entirely determined by how γ maps elements to the support of g. We use this result to show that the evaluation of g for an input structure $\mathcal{A}$ is characterised by a set EV_g of assignments to the (constant-size) support of g. We use the various algorithms in Chapter 7, as well as numerous other constructions given throughout this chapter, to recursively construct EV_g for each gate g. Finally, we use the fixed-point operator to implement this recursive definition and, by evaluating the output gates of the circuit, we define the required formula.

This approach reduces to proving four key claims. First, that there is a constant bound on the size of the supports of the gates (and elements of the universes of the gates) in each circuit. Second, that each circuit can be mapped in polynomial-time to an equivalent circuit with unique labels (this is needed to apply the support theorem). Third, that the orbits and supports of the gates in each circuit can be computed in polynomial-time. Fourth, that there

exists a formula in the logic that recursively defines EV_g for each gate g in each circuit. We have already proved the first three of these claims. It remains for us to discuss the fourth claim.

Anderson and Dawar [3] prove this claim for symmetric circuits with majority gates by first exhibiting a bijection from the orbit of a gate to the set of assignments to its support and then, using this bijection, counting the number of children of g that evaluate to true. This suffices as the gates in the circuits of interest to them all compute trivially invariant functions, and so counting the number of inputs that evaluate to true suffices to evaluate a gate. However, in our more general setting, the gates in a circuit compute possibly non-trivially invariant functions, and the evaluation of such a gate depends not just on the *number* of children that evaluate to true, but on the *structure* defined at the gate g.

In this chapter we develop a novel approach for recursively defining EV_g for each gate g. In particular, we show that for each gate g and each assignment to the support of g we can define a structure $M_{\equiv}^{g,\eta}$ isomorphic to the structure defined at g. We then evaluate g in the logic by applying the relevant generalised operator to the interpretation defining $M_{\equiv}^{g,\eta}$.

We should note that Proposition 8.1 generalises the translation from P-uniform families of symmetric circuits with *majority gates* to FPC proved by Anderson and Dawar [3]. The proof they present is broadly similar in structure to the proof we present in this chapter and can similarly be reduced to proving four analogous claims. However, we should emphasise that in each of these cases Anderson and Dawar's arguments crucially rely on the symmetry assumption on gates, and substantial new developments were needed to prove these results in the more general setting.

This chapter is organised as follows. In the first section we show how to construct a structure $M_{\equiv}^{g,\eta}$ for each gate g, appropriate structure $\mathcal{A}$, and assignment η to the support of g. We also show that $M_{\equiv}^{g,\eta}$ is isomorphic to the structure defined at g when evaluating the circuit for the input $\mathcal{A}$. In the second section we complete the proof of Proposition 8.1 by showing that the construction given in the first section can be implemented as an interpretation in $\mathrm{FP}^{\mathbb{N}}$.

8.1 Defining a Structure at Each Gate

We now show that for a gate g in a symmetric circuit C of order n the evaluation of g for an input structure $\mathcal{A}$ and bijection $\gamma \in [n]^{\underline{A}}$ depends only on the mapping given by γ to the support of g. We first introduce some notation. We say two injective functions are *compatible* if we can define an injection on the union of their domains that agrees with both functions on their respective domains. We now define this notion more formally and more generally.

Definition 8.2. Let X_1, X_2, Y_1, Y_2 be sets. We say that $f : X_1 \to Y_1$ and $g : X_2 \to Y_2$ are *compatible* if (i) for all $x \in X_1 \cap X_2$ we have $f(x) = g(x)$, (ii) for all $x \in X_1 \setminus X_2$ and $y \in X_2$ we have $f(x) \neq g(y)$, and (iii) for all $x \in X_2 \setminus X_1$ and $y \in X_1$ we have $g(x) \neq f(y)$.

Lemma 8.3. *Let ρ be a vocabulary and $\mathbb{B}$ a basis, and C be a symmetric $(\mathbb{B}, \rho)$-circuit with unique labels. Let $n \in \mathbb{N}$ be the order of C and let $\mathcal{A} \in \mathrm{fin}[\rho, n]$. Let g be an internal gate, $\eta \in A^{\underline{sp(g)}}$, and $\gamma_1, \gamma_2 \in [n]^{\underline{A}}$ such that $\gamma_1^{-1} \sim \eta$ and $\gamma_2^{-1} \sim \eta$. Then $L^{\gamma_1\mathcal{A}}(g)$ and $L^{\gamma_2\mathcal{A}}(g)$ are isomorphic and $C[\gamma_1\mathcal{A}](g) = C[\gamma_2\mathcal{A}](g)$.*

Proof. We have that there exists a unique $\pi \in \mathbf{Sym}_n$ such that $\pi\gamma_1 = \gamma_2$. Moreover, since γ_1^{-1} and γ_2^{-1} are both compatible with η, it follows that π must fix $\mathrm{sp}(g)$ pointwise. From the definition of a support, we have that $\pi(g) = g$, and so $L(g)$ is isomorphic to $\pi L(g)$. Therefore there exists $\lambda \in \mathbf{Aut}(g)$ such that $\pi L(g) = L(g)\lambda$, and so for all $a \in \mathrm{ind}(g)$,

$$\begin{aligned} L^{\gamma_1\mathcal{A}}(g)(a) &= C[\gamma_1\mathcal{A}](L(g)(a)) \\ &= C[\pi\gamma_1\mathcal{A}][\pi L(g)(a)] \\ &= C[\gamma_2\mathcal{A}][L(g)(\lambda(a))] \\ &= L^{\gamma_2\mathcal{A}}(g)(\lambda(a)). \end{aligned}$$

It follows that $L^{\gamma_1\mathcal{A}}(g)$ and $L^{\gamma_2\mathcal{A}}(g)$ are isomorphic and $C[\gamma_1\mathcal{A}](g) = \Sigma(g)(L^{\gamma_1\mathcal{A}}(g)) = \Sigma(g)(L^{\gamma_2\mathcal{A}}(g)) = C[\gamma_2\mathcal{A}](g)$. □

For the remainder of this section we fix a basis $\mathbb{B}$ such that there exists $r_{\mathbb{B}} \in \mathbb{N}$ such that every relation in the vocabulary of a structured function in $\mathbb{B}$ has arity at most $r_{\mathbb{B}}$. Let ρ be a relational vocabulary. We fix a P-uniform family of transparent symmetric $(\mathbb{B}, \rho)$-circuits $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$. Let n_0 and k be the constants in the statement of Theorem 6.24. Let $n_1 = \max(n_0, k \cdot r_{\mathbb{B}})$. Fix some $n > n_1$ and $\mathcal{A} \in \mathrm{fin}[\rho, n]$. Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle := C_n$.

For each $g \in G$ let $\Gamma_g := \{\gamma \in [n]^{\underline{A}} : C[\gamma\mathcal{A}](g) = 1\}$ and let $\mathrm{EV}_g := \{\eta \in A^{\underline{\mathrm{sp}(g)}} : \exists\gamma \in \Gamma_g\,, \eta \sim \gamma^{-1}\}$. In other words, Γ_g is the set of bijections for which g evaluates to one and EV_g is the set of assignments to the support of g that can be extended to a bijection for which g evaluates to one. It follows from Lemma 8.3 that Γ_g is entirely determined by EV_g. It is important to note that, from the support theorem, the domain of each $\eta \in \mathrm{EV}_g$ has cardinality at most k. As such, we think of EV_g as succinctly encoding Γ_g and characterising the evaluation of the gate g.

We aim to define the set EV_g for each gate g in C by induction on the structure of the circuit. We do this by inductively defining for each gate $g \in \mathrm{EV}_g$ and each assignment $\alpha \in A^{\underline{\mathrm{sp}(g)}}$ a structure $\mathcal{M}_{\equiv}^{g,\eta}$ such that for any $\gamma \in [n]^{\underline{A}}$ with $\gamma^{-1} \sim \eta$, $M_{\equiv}^{g,\eta}$ is isomorphic to $L^{\gamma\mathcal{A}}(g)$. In this section we give an inductive definition of $\mathcal{M}_{\equiv}^{g,\eta}$ and in Section 8.2 we show that each step in this definition can be implemented by a formula in FPC and hence $\mathcal{M}_{\equiv}^{g,\eta}$ is definable by an FPC-interpretation in $\mathcal{A}$.

For the remainder of this section we fix a gate $g \in G$ and an assignment $\eta \in A^{\underline{\mathrm{sp}(g)}}$. Let $\tau := (\boldsymbol{R}, \boldsymbol{S}, \zeta)$ be the vocabulary of g. Let $X = \uplus_{s \in \boldsymbol{S}} X_s$ be the universe of g. We now introduce some notation. Let $h \in H_g$. There exists $(\vec{x}, R) \in \mathrm{ind}(g)$ such that $L(g)(\vec{x}, R) = h$ and for each $j \in [r_R]$ we let $\mathrm{col}_j(h)$ denote the jth element in the tuple $\vec{x}$. We follow the conventions introduced in Chapter 6 and for each $x \in \mathrm{unv}(g)$ we shorten some notation and let $\mathrm{sp}(x) := \mathrm{sp}_{\mathbf{Stab}(\mathrm{sp}(g))}(x)$, $\mathbf{Orb}(x) := \mathbf{Orb}_{\mathbf{Stab}(\mathrm{sp}(g))}(x)$, and $\mathbf{Stab}(x) := \mathbf{Stab}_{\mathbf{Stab}(\mathrm{sp}(g))}(x)$. For each $h \in H_g$ let $A^h := \{\alpha \in A^{\underline{\mathrm{sp}(h)}} : \eta \sim \alpha\}$ be the set of assignments to the support of h that are compatible with η. For each $x \in \mathrm{unv}(g)$ let $A^x := \{\alpha \in A^{\underline{\mathrm{sp}(x)}} : \eta \sim \alpha\}$ be the set of assignments to the support of x that are compatible with η.

We first aim to define a τ-structure $M^{g,\eta}$ and then show that there is an epimorphism from $L^{\gamma\mathcal{A}}(g)$ to $M^{g,\eta}$. We can define $M^{g,\eta}_{\equiv}$ by quotienting $M^{g,\eta}$ by the equivalence relation on $M^{g,\eta}$ induced by the surjection, and hence establish the isomorphism. We now define the set $I^{g,\eta}$ and then define the τ-structure $M^{g,\eta}$ with universe $I^{g,\eta}$.

We note that in Section 8.2 we encode each circuit as a structure over an ordered universe and this order induces an order on the universe of each gate. As such, in this section we suppose for each $s \in \boldsymbol{S}$ that there is some linear ordering on X_s. For each $s \in \boldsymbol{S}$ let $\mathrm{minorb}(s) := \{\min(\mathbf{Orb}(x)) : x \in X_s\}$ and let $I^{g,\eta}_s := \{(x, \alpha) : x \in \mathrm{minorb}(s), \alpha \in A^x\}$. Let $I^{g,\eta} := \uplus_{s \in \boldsymbol{S}} I^{g,\eta}_s$. For each $s \in \boldsymbol{S}$ we think of each $x \in \mathrm{minorb}(s)$ as denoting an orbit in X_s and each assignment in A^x as encoding a permutation on $\mathrm{sp}(x)$ and hence an element of the orbit. In this way we can think of each element in $I^{g,\eta}_s$ as encoding an element in X_s.

We now sketch the construction of $M^{g,\eta}$ and then complete this construction formally below. We aim to associate each $((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R \in \tau[I^{g,\eta}]$ with a gate $h \in H_g$ and assignment $\epsilon \in A^h$. In order to define this pair we first define a set of permutations $\sigma_1, \ldots, \sigma_{r_R} \in \mathbf{Stab}(\mathrm{sp}(g))$ such that the action of each σ_i on α_i for $i \in [r_R]$ defines a set of compatible assignments. We let ϵ be the injection defined on the union of these compatible assignments and let $h := L(g)((x_1, \ldots, x_{r_R}), R)$. We then define $M^{g,\eta}$ by letting $M(((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R) = 1$ if, and only if, $\epsilon \in \mathrm{EV}_h$. We complete this definition formally.

We first define a function $\bar{J}^{g,\eta}$ that maps each element in $\tau[I^{g,\eta}]$ to a sequence of injections as follows. Let $z := (((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R) \in \tau[I^{g,\eta}]$. Let $\bar{J}^{g,\eta}(z) := (\bar{\sigma}_1, \ldots, \bar{\sigma}_{r_R})$, where for each $j \in [r_R]$ have that $\bar{\sigma}_j \in [n]^{\underline{\mathrm{sp}(x_j)}}$ is defined by recursion as follows. For each $u \in \mathbf{Dom}(\alpha_1)$ let $\bar{\sigma}_1(u) := u$ if $u \in \mathrm{sp}(g)$ and otherwise let $\bar{\sigma}_1(u)$ be the k_uth element of $[n] \setminus \mathrm{sp}(g)$, where k_u is such that u is the k_uth element of $\mathbf{Dom}(\alpha_1) \setminus \mathrm{sp}(g)$. Suppose $j > 1$ and let $u \in \mathbf{Dom}(\alpha_j)$. If $u \in \mathrm{sp}(g)$ let $\bar{\sigma}_j(u) := u$. If $u \notin \mathrm{sp}(g)$ and there exists some minimal $j' < j$ and $u' \in \mathbf{Dom}(\alpha_{j'})$ such that $\alpha_j(u) = \alpha_{j'}(u')$ then let $\bar{\sigma}_j(u) := \bar{\sigma}_{j'}(u')$. Otherwise, let $\bar{\sigma}_j(u)$ be the $((j-1)k + k_u)$th element of $[n] \setminus \mathrm{sp}(g)$, where k_u is such that u is the k_uth element of $\mathbf{Dom}(\alpha_j) \setminus \mathrm{sp}(g)$.

The construction ensures that for each $R \in \boldsymbol{R}$ and $i \in [r_R]$ we have that $\bar{\sigma}_i$ is an injection. We abuse notation slightly and write $\alpha_i \bar{\sigma}_i^{-1}$ to denote the injection $\alpha_i \bar{\sigma}_i^{-1} : \mathbf{Img}(\bar{\sigma}_i) \to A$

defined by $\alpha_i\bar{\sigma}_i^{-1}(u) = \alpha_i(\bar{\sigma}_i^{-1}(u))$ for all $u \in \mathbf{Img}(\bar{\sigma}_i)$. We think of each $\alpha_i\bar{\sigma}_i^{-1}$ as being a version of α_i with the domain shifted by $\bar{\sigma}_i^{-1}$ such that the set $\{\alpha_i\bar{\sigma}_i^{-1} : i \in [r_R]\}$ consists of pairwise compatible assignments. Moreover, we note that $\mathrm{sp}(g) \subseteq \mathbf{Dom}(\bar{\sigma}_i)$ and for each $a \in \mathrm{sp}(g)$ we have $\bar{\sigma}_i(a) = a$. It follows that each $\bar{\sigma}_i$ can be extended to a permutation in $\mathbf{Stab}(\mathrm{sp}(g))$. We now formally prove each of the assertions we have just made.

Lemma 8.4. *Let $z = (((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R) \in \tau[I^{g,\eta}]$. Let $(\bar{\sigma}_1, \ldots, \bar{\sigma}_{r_R}) := \bar{J}^{g,\eta}(z)$. Then for all $j \in [r_R]$ we have that*

1. *for all $u \in sp(g) \cap \boldsymbol{Dom}(\bar{\sigma}_j)$, $\bar{\sigma}_j(u) = u$,*
2. *$\bar{\sigma}_j$ is an injection, and*
3. *for all $j' \in [r_R]$, $\alpha_j \circ \bar{\sigma}_j^{-1}$ and $\alpha_{j'} \circ \bar{\sigma}_{j'}^{-1}$ are compatible.*

Proof. It is easy to see that for all $j \in [r_R]$ we have for all $u \in \mathrm{sp}(g) \cap \mathbf{Dom}(\bar{\sigma}_j)$ that $\bar{\sigma}_j(u) = u$. We now prove by induction that for all $j \in [r_R]$, $\bar{\sigma}_j$ is an injection and for all $j' < j$, $\alpha_j \circ \bar{\sigma}_j^{-1}$ and $\alpha_{j'} \circ \bar{\sigma}_{j'}^{-1}$ are compatible.

Let $j \in [r_R]$. Suppose $j = 1$. Let $u, v \in \mathbf{Dom}(\alpha_1)$ be such that $\bar{\sigma}_1(u) = \bar{\sigma}_1(v)$. Suppose $u \in \mathrm{sp}(g)$. Then $v \in \mathrm{sp}(g)$, as if $v \notin \mathrm{sp}(g)$ then, from the definition of $\bar{\sigma}_1$, we have $\bar{\sigma}_1(v) \notin \mathrm{sp}(g)$, but $\bar{\sigma}_1(v) = \bar{\sigma}_1(u) = u \in \mathrm{sp}(g)$. It follows that $u = \bar{\sigma}_1(u) = \bar{\sigma}_1(v) = v$. Otherwise, suppose $u \notin \mathrm{sp}(g)$. Then $v \notin \mathrm{sp}(g)$. We have $\bar{\sigma}_1(u)$ is the k_uth element of $[n] \setminus \mathrm{sp}(g)$, where k_u is such that u is the k_uth element of $\mathbf{Dom}(\alpha_1) \setminus \mathrm{sp}(g)$ and $\bar{\sigma}_1(v)$ is the k_vth element of $[n] \setminus \mathrm{sp}(g)$ where k_v is such that v is the k_vth element of $\mathbf{Dom}(\alpha_1) \setminus \mathrm{sp}(g)$. But $\bar{\sigma}_1(u) = \bar{\sigma}_1(v)$ and so $k_u = k_v$ and $u = v$. It follows that $\bar{\sigma}_1$ is an injection. This completes the proof of the base case.

Suppose $j > 1$ and suppose the induction hypothesis holds for all $j' < j$. It can be shown, following a very similar approach as for the base case, that $\bar{\sigma}_j$ is an injection. Let $j' < j$. We first prove a claim.

Claim 8.4.1. *Let $u \in \boldsymbol{Dom}(\alpha_j\bar{\sigma}_j^{-1})$ and $u' \in \boldsymbol{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$. Then $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u'))$ if, and only if, $u = u'$.*

Proof. '$\Rightarrow$' Suppose $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u'))$. If $u \in \mathrm{sp}(g)$ then $\eta(u') = \alpha_{j'}(u') = \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u')) = \alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_j(u) = \eta(u)$ and, since η is an injection, it follows that $u = u'$. Suppose $u \notin \mathrm{sp}(g)$. Since $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u'))$, it follows that there exists a minimal $j'' < j$ and $u'' \in \mathbf{Dom}(\alpha_{j''})$ such that $\alpha_j\bar{\sigma}_j^{-1}(u) = \alpha_{j'}\bar{\sigma}_{j'}^{-1}(u') = \alpha_{j''}(u'')$. Then $u = \bar{\sigma}_j(\bar{\sigma}_j^{-1}(u)) = \bar{\sigma}_{j''}(u'') = \bar{\sigma}_{j'}(\bar{\sigma}_{j'}^{-1}(u')) = u'$.

'$\Leftarrow$' Suppose $u = u'$. Suppose that both $u \notin \mathrm{sp}(g)$ and for all $j'' < j$ and all $u'' \in \mathbf{Dom}(\alpha_{j''})$ we have $\alpha_j(\bar{\sigma}_j^{-1}(u)) \neq \alpha_{j''}(u'')$. Then $u > u_1$, where u_1 is the $(j-1)k$th element in $[n] \setminus \mathrm{sp}(g)$. Let $j'' < j$ and $u'' \in \mathbf{Dom}(\alpha_{j''})$. If $u'' \in \mathrm{sp}(g)$ then $u \neq u'' = \bar{\sigma}_{j''}(u'')$. Otherwise $u'' \notin \mathrm{sp}(g)$ and it can be shown that $\bar{\sigma}_{j''}(u'') \leq u_2 \leq u_1$, where u_2 is the $((j''-1)k + k)$th element of $[n] \setminus \mathrm{sp}(g)$. It follows that $\bar{\sigma}_{j''}(u'') \leq u_1 < a$ and so

$u \neq \bar{\sigma}_{j''}(u'')$. Thus, for all $j'' \leq j$ we have $u \notin \mathbf{Img}(\bar{\sigma}_{j''})$. Then $u \notin \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$. But this is a contradiction as $u = u' \in \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$.

It follows that either $u \in \mathrm{sp}(g)$ or there exists $j'' < j$ and $u'' \in \mathbf{Dom}(\alpha_{j''})$ such that $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j''}(u'')$. If $u \in \mathrm{sp}(g)$ then $\bar{\sigma}_j(u) = u = u' = \bar{\sigma}_{j'}(u')$ and so $\alpha_j\bar{\sigma}_j^{-1}(u) = \alpha_j(u) = \eta(u) = \eta(u') = \alpha_{j'}(u') = \alpha_{j'}\bar{\sigma}_{j'}^{-1}(u')$. Suppose $u \notin \mathrm{sp}(g)$. Let $j'' < j$ be minimal such that there exists $u'' \in \mathbf{Dom}(\alpha_{j''})$ such that $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j''}(u'')$. Then $u' = u = \bar{\sigma}_j(\bar{\sigma}_j^{-1}(u)) = \bar{\sigma}_{j'}(u'')$. It follows that $u'' \in \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$. From the induction hypothesis we have that $\alpha_{j'}\bar{\sigma}_{j'}^{-1}$ and $\alpha_{j''}\bar{\sigma}_{j''}^{-1}$ are compatible, and so $\alpha_{j'}\bar{\sigma}_{j'}^{-1}(u') = \alpha_{j''}\bar{\sigma}_{j''}^{-1}(\bar{\sigma}_{j''}(u'')) = \alpha_{j''}(u'') = \alpha_j(\bar{\sigma}_j^{-1}(u))$.

This completes the proof of Claim 8.4.1. □

Let $u \in \mathbf{Dom}(\alpha_j\bar{\sigma}_j^{-1}) \cap \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$. Then, from Claim 8.4.1, it follows that $\alpha_j(\bar{\sigma}_j^{-1}(u)) = \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u))$. Let $u \in \mathbf{Dom}(\alpha_j\bar{\sigma}_j^{-1}) \setminus \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1})$ and $u' \in \mathbf{Dom}(\alpha_{j'}\bar{\sigma}_{j'}^{-1}) \setminus \mathbf{Dom}(\alpha_j\bar{\sigma}_j^{-1})$. Then, from Claim 8.4.1, since $u \neq u'$ it follows $\alpha_j(\bar{\sigma}_j^{-1}(u)) \neq \alpha_{j'}(\bar{\sigma}_{j'}^{-1}(u'))$. We conclude that $\alpha_j\bar{\sigma}_j^{-1}$ and $\alpha_{j'}\bar{\sigma}_{j'}^{-1}$ are compatible. This completes the proof of the lemma. □

Let $J^{g,\eta} : \tau[I^{g,\eta}] \to \{(h,\epsilon) : h \in H_g, \epsilon \in A^h\}$ be defined for $z := (((x_1,\alpha_1),\ldots,(x_{r_R},\alpha_{r_R})),R) \in \tau[I^{g,\eta}]$ as follows. Let $(\bar{\sigma}_1,\ldots,\bar{\sigma}_{r_R}) := \bar{J}^{g,\eta}(z)$. For every $j \in [r_R]$ let $\sigma_j \in \mathbf{Stab}(\mathrm{sp}(g))$ be such that for all $u \in \mathrm{sp}(x_j)$, $\sigma_j(u) = \bar{\sigma}_j(u)$. Let $h := L(g)((\sigma(x_1),\ldots,\sigma(x_{r_R})),R)$ and let $\epsilon := (\alpha_1\bar{\sigma}_1^{-1}|\ldots,|\alpha_{r_R}\bar{\sigma}_{r_R}^{-1})\Big|_{\mathrm{sp}(h)}$. It follows from Lemma 8.4 that the assignments $\alpha_1\bar{\sigma}_1^{-1},\ldots,\alpha_{r_R}\bar{\sigma}_{r_R}^{-1}$ are pair-wise compatible and so ϵ is well-defined. Let $J^{g,\eta}(z) := (h,\epsilon)$.

Let $M^{g,\eta} : \tau[I^{g,\eta}] \to \{0,1\}$ be defined for $z := (((x_1,\alpha_1),\ldots,(x_{r_R},\alpha_{r_R})),R) \in \tau[I^{g,\eta}]$ as follows. Let $(h,\epsilon) := J^{g,\eta}(z)$ and let $M^{g,\eta}(z) = 1$ if, and only if, $\epsilon \in \mathrm{EV}_h$.

We previously stated that we can think of each assignment to the support of a child or element of the universe of g as encoding an element in the orbit of that object. We now formalise this statement and show that for each $\gamma \in [n]^{\underline{A}}$ with $\gamma^{-1} \sim \eta$ and each $h \in H_g$ there is a natural surjective map defined by γ from A^h to $\mathbf{Orb}(h)$ and for each $x \in \mathrm{unv}(g)$ there is a similarly defined map from A^x to $\mathbf{Orb}(x)$. Let $\gamma \in [n]^{\underline{A}}$ be such that $\gamma^{-1} \sim \eta$. Let $h \in H_g$. For each $\epsilon \in A^h$ let Π_ϵ^γ be any permutation in $\mathbf{Stab}(\mathrm{sp}(g))$ such that $\Pi_\epsilon^\gamma(u) = \gamma(\epsilon(u))$ for all $u \in \mathrm{sp}(h)$. It follows from Lemma 6.14 that the action of Π_ϵ^γ on h is defined independently of this choice of permutation and so the function $\epsilon \mapsto \Pi_\epsilon^\gamma(h)$ is well-defined. Let $x \in X$. For each $\alpha \in A^x$ let Π_α^γ be any permutation in $\mathbf{Stab}(\mathrm{sp}(g))$ such that $\Pi_\alpha^\gamma(u) = \gamma(\alpha(u))$ for all $u \in \mathrm{sp}(x)$. It follows similarly that the action of Π_α^γ on x is defined independently of the choice of permutation and so the function $\alpha \mapsto \Pi_\alpha^\gamma(x)$ is well-defined. It is easy to show that both of these maps are surjective.

We now define a function from $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$. Let $\gamma \in [n]^{\underline{A}}$ be such that $\gamma^{-1} \sim \eta$. For each $s \in \boldsymbol{S}$ let $P_s^\gamma : I_s^{g,\eta} \to X_s$ be defined by $P_s^\gamma(x,\alpha) := \Pi_\alpha^\gamma(x)$ for all $(x,\alpha) \in I_s^{g,\eta}$. Let $P^\gamma = \uplus_{s\in\boldsymbol{S}} P_s^\gamma$. We aim to show that P^γ defines an epimorphism from $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$. We

will prove this result in stages. We first establish a correspondence between EV_h and those elements of the orbit of h that evaluate to 1 in the circuit.

Lemma 8.5. *Let $\gamma \in [n]^{\underline{A}}$ be such that $\gamma^{-1} \sim \eta$ and $h \in H_g$. Then $\epsilon \in EV_h$ if, and only if, $C[\gamma\mathcal{A}](\Pi^{\gamma}_{\epsilon}(h)) = 1$.*

Proof. From the definition of EV_h it follows that $\epsilon \in \mathrm{EV}_h$ if, and only if, there exists $\delta_\epsilon \in [n]^{\underline{A}}$ such that $\delta_\epsilon^{-1} \sim \eta$, $\delta_\epsilon^{-1} \sim \epsilon$, and $C[\delta_\epsilon\mathcal{A}](h) = 1$. Let $\pi_\epsilon := \gamma\delta_\epsilon^{-1}$. Notice that $\pi \in \mathbf{Stab}(\mathrm{sp}(g))$ and $\pi_\epsilon\delta_\epsilon = \gamma$. For $a \in \mathrm{sp}(h)$ we have $\gamma(\epsilon(a)) = \pi_\epsilon(\delta_\epsilon(\epsilon(a))) = \pi_\epsilon(a)$ (the second equality follows from the fact that $\delta_\epsilon^{-1} \sim \epsilon$ and so $\delta_\epsilon(\epsilon(a)) = a$). It follows that for all $a \in \mathrm{sp}(h)$ we have $\Pi^{\gamma}_{\epsilon}(a) = \gamma(\epsilon(a)) = \pi_\epsilon(a)$, and so $\Pi^{\gamma}_{\epsilon}(h) = \pi_\epsilon(h)$. Thus

$$\begin{aligned}\epsilon \in \mathrm{EV}_h \iff C[\delta_\epsilon\mathcal{A}](h) = 1 &\iff C[\pi_\epsilon\delta_\epsilon\mathcal{A}](\pi_\epsilon(h)) = 1 \\ &\iff C[\gamma\mathcal{A}](\pi_\epsilon(h)) = 1 \iff C[\gamma\mathcal{A}](\Pi^{\gamma}_{\epsilon}(h)) = 1.\end{aligned}$$

□

We recall that the function $J^{g,\eta}$ maps tuples of the form $(((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R)$ to pairs of the form (h, ϵ), where h is a gate defined by mapping each $x_1, \ldots, x_{r_R}$ such that each of the images of the assignments $\alpha_1, \ldots, \alpha_{r_R}$ under the same mapping form a set of pairwise compatible assignments and ϵ is the assignment to the support of h given by taking the union of these compatible assignments. We now show that $\Pi^{\gamma}_{\epsilon}(h)$ is exactly the gate defined by mapping each x_i to $\Pi^{\gamma}_{\alpha_i}$.

Lemma 8.6. *Let $\gamma \in [n]^{\underline{A}}$ be such that $\gamma^{-1} \sim \eta$. Let $z := (((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R) \in \tau[I^{g,\eta}]$ and let $(h, \epsilon) := J^{g,\eta}(z)$. Then $\Pi^{\gamma}_{\epsilon}(h) = L(g)(((\Pi^{\gamma}_{\alpha_1}(x_1), \ldots, \Pi^{\gamma}_{\alpha_{r_R}}(x_{r_R}), R))$.*

Proof. Let $j \in [r_i]$ and let $u \in \mathrm{sp}(x_j)$. It follows from Theorem 6.24 that $\sigma_j(u) \in \mathrm{sp}(\mathrm{col}_j(h)) \subseteq \mathrm{sp}(h) \cup \mathrm{sp}(g)$, and so either $\sigma_j(u) \in \mathrm{sp}(g)$ or $\sigma_j(u) \in \mathrm{sp}(h)$. Suppose $\sigma_j(u) \in \mathrm{sp}(g)$. We have that σ_j and Π^{γ}_{ϵ}, and $\Pi^{\gamma}_{\alpha_j}$ are in $\mathbf{Stab}(\mathrm{sp}(g))$, and so $\Pi^{\gamma}_{\epsilon}(\sigma_j(u)) = u = \Pi^{\gamma}_{\alpha_j}(u)$. Suppose instead that $\sigma_j(u) \in \mathrm{sp}(h)$. Then $\Pi^{\gamma}_{\epsilon}(\sigma_j(u)) = \gamma(\epsilon(\sigma_j(u))) = \gamma(\alpha_j(\bar{\sigma}_j^{-1}(\sigma_j(u)))) = \gamma(\alpha_j(u)) = \Pi^{\gamma}_{\alpha_j}(u)$. It follows that $\Pi^{\gamma}_{\epsilon}(\sigma_j(x_j)) = \Pi^{\gamma}_{\alpha_j}(x_j)$ and so

$$\begin{aligned}\Pi^{\gamma}_{\epsilon}(h) &= \Pi^{\gamma}_{\epsilon}(L(g)((\sigma_1(x_1), \ldots, \sigma_{r_i}(x_{r_i})), R_i)) \\ &= L(g)((\Pi^{\gamma}_{\epsilon}(\sigma_1(x_1)), \ldots, \Pi^{\gamma}_{\epsilon}(\sigma_{r_i}(x_{r_i}))), R_i) \\ &= L(g)((\Pi^{\gamma}_{\alpha_1}(x_1), \ldots, \Pi^{\gamma}_{\alpha_{r_i}}(x_{r_i})), R_i)\end{aligned}$$

□

We now show that P^γ is a homomorphism from $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$ for any $\gamma \in [n]^{\underline{A}}$ such that $\gamma^{-1} \sim \eta$. This result is an immediate consequence of Lemmas 8.5 and 8.6.

Proposition 8.7. *Let* $z := (((x_1, \alpha_1), \ldots, (x_{r_R}, \alpha_{r_R})), R) \in \boldsymbol{Dom}(M^{g,\eta})$. *Let* $\gamma \in [n]^{\underline{A}}$ *be such that* $\gamma^{-1} \sim \eta$. *Then* $M^{g,\eta}(z) = L^{\gamma\mathcal{A}}(g)(P^\gamma(x_1, \alpha_1), \ldots, P^\gamma(x_{r_R}, \alpha_{r_R})), R)$.

Proof. Let $(h, \epsilon) := J^{g,\eta}(z)$. Then

$$\begin{aligned} M^{g,\eta}(z) = 1 &\iff \epsilon \in \mathrm{EV}_h \\ &\iff C[\gamma\mathcal{A}](\Pi^\gamma_\epsilon(h)) = 1 \\ &\iff C[\gamma\mathcal{A}](L(g)((\Pi^\gamma_{\alpha_1}(x_1), \ldots, \Pi^\gamma_{\alpha_{r_R}}(x_{r_R})), R)) = 1 \\ &\iff L^{\gamma\mathcal{A}}(g)(P^\gamma(x_1, \alpha_1), \ldots, P^\gamma(x_{r_R}, \alpha_{r_R})), R) = 1. \end{aligned}$$

The second equivalence follows from Lemma 8.5 and the third equivalence follows from Lemma 8.6. □

We now show that P^γ is a surjection. This follows almost immediately from the fact that for any $y \in X$ the mapping $\alpha \mapsto \Pi^\gamma_\alpha(y)$ for any $\alpha \in A^y$ is a surjection.

Lemma 8.8. *If* $\gamma \in [n]^{\underline{A}}$ *is such that* $\gamma^{-1} \sim \eta$ *then* P^γ *is surjective.*

Proof. Let $x \in X$. Let $y = \min(\mathbf{Orb}(x))$. There exists $\sigma \in \mathbf{Stab}(\mathrm{sp}(g))$ such that $\sigma(y) = x$. Let $\alpha := \gamma^{-1}\sigma\big|_{\mathrm{sp}(y)}$. Then for each $u \in \mathrm{sp}(y) \cap \mathrm{sp}(g)$ we have $\alpha(u) = \gamma^{-1}\sigma\big|_{\mathrm{sp}(y)}(u) = \gamma^{-1}(u) = \eta(u)$, and so $\alpha \in A^y$. For each $u \in \mathrm{sp}(y)$ we have $\Pi^\gamma_\alpha(u) = \gamma(\alpha(u)) = \gamma(\gamma^{-1}\sigma\big|_{\mathrm{sp}(y)}(u)) = \sigma(u)$. It follows that $P^\gamma(y, \alpha) = \Pi^\gamma_\alpha(y) = \sigma(y) = x$. □

It follows from Lemma 8.8 and Proposition 8.7 that for any $\gamma \in [n]^{\underline{A}}$ such that $\gamma^{-1} \sim \eta$, P^γ is an epimorphism from $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$. We aim to establish the existence of an isomorphism from a quotient of $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$. We now define an equivalence relation on $I^{g,\eta}$ and show that it is a congruence on $M^{g,\eta}$. We then show that this equivalence relation identifies precisely those elements in $I^{g,\eta}$ that are mapped to the same element by P^γ. We then use the first isomorphism theorem to establish the existence of the isomorphism between the quotient of $M^{g,\eta}$ by the equivalence relation and $L^{\gamma\mathcal{A}}(g)$.

Definition 8.9. We say that $(x, \alpha), (y, \beta) \in I^{g,\eta}$ are *mutually stable* if $x = y$ and there exists $\pi \in \mathbf{Stab}(x)$ such that $\alpha(u) = \beta(\pi(u))$ for all $u \in \mathrm{sp}(x)$. We write $(x, \alpha) \equiv (y, \beta)$ to denote that (x, α) and (y, β) are mutually stable.

We now show that two pairs are mutually stable if, and only if, they are mapped to the same element by P^γ.

Lemma 8.10. *Let* $\gamma \in [n]^{\underline{A}}$ *be such that* $\gamma^{-1} \sim \eta$. *For each* $(x, \alpha), (y, \beta) \in I^{g,\eta}$ *we have that* $(x, \alpha) \equiv (y, \beta)$ *if, and only if,* $P^\gamma(x, \alpha) = P^\gamma(y, \beta)$.

Proof. '⇒' Let $(x, \alpha), (y, \beta) \in I^{g,\eta}$ and suppose $(x, \alpha) \equiv (y, \beta)$. Since $(x, \alpha) \equiv (y, \beta)$ there exists $\pi \in \mathbf{Stab}(x)$ such that for all $u \in \mathrm{sp}(x)$, $\alpha(u) = \beta(\pi(u))$ and $\pi(x) = x =$

$y = \pi(y)$. For all $u \in \mathbf{Stab}(x)$ we have $\Pi^\gamma_\beta(\pi(u)) = \gamma(\beta(\pi(u))) = \gamma(\alpha(u)) = \Pi^\gamma_\alpha(u)$, and so $\Pi^\gamma_\beta(\pi(x)) = \Pi^\gamma_\beta(\pi(y)) = \Pi^\gamma_\alpha(y) = \Pi^\gamma_\alpha(x)$. It follows that $P^\gamma(y,\beta) = \Pi^\gamma_\beta(y) = \Pi^\gamma_\beta(\pi(y)) = \Pi^\gamma_\alpha(x) = P^\gamma(x,\alpha)$.

'$\Leftarrow$' Let $(x,\alpha),(y,\beta) \in I^{g,\eta}$ be such that $P^\gamma(x,\alpha) = P^\gamma_\equiv(y,\beta)$. Then $\Pi^\gamma_\alpha(x) = \Pi^\gamma_\beta(y)$. It follows that $x \in \mathbf{Orb}(y)$, and so $\mathbf{Orb}(x) = \mathbf{Orb}(y)$ and $x = \min(\mathbf{Orb}(x)) = \min(\mathbf{Orb}(y)) = y$. Let $\sigma := (\Pi^\gamma_\beta)^{-1}\Pi^\gamma_\alpha$. Notice that $\sigma(x) = (\Pi^\gamma_\beta)^{-1}\Pi^\gamma_\alpha(x) = y = x$, and so $\sigma \in \mathbf{Stab}(x)$. Let $u \in \mathrm{sp}(x)$. Then $\gamma(\alpha(u)) = \Pi^\gamma_\alpha(u) = \Pi^\gamma_\beta(\sigma(u)) = \gamma(\beta(\sigma(u)))$. Since γ is an injection it follows that $\alpha(u) = \beta(\sigma(u))$ and so $(x,\alpha) \equiv (y,\beta)$.

□

We now show that mutual stability is a congruence with respect to $M^{g,\eta}$. This result follows from the fact that mutual stability is a congruence with respect to P^γ and the fact that P^γ is a homomorphism.

Lemma 8.11. *Let* $(((x_1,\alpha_1),\ldots,(x_{r_R},\alpha_{r_R})),R),(((y_1,\beta_1),\ldots,(y_{r_{r_T}},\beta_{r_{r_T}})),T) \in \tau[I^{g,\eta}]$. *Suppose* $R = T$ *and for all* $i \in [r_R]$ *we have* $(x_i,\alpha_i) \equiv (y_i,\beta_i)$. *It follows that*

$$M^{g,\eta}(((x_1,\alpha_1),\ldots,(x_{r_R},\alpha_{r_R})),R) = M^{g,\eta}(((y_1,\beta_1),\ldots,(y_{r_R},\beta_{r_T})),T).$$

Proof. We have

$$\begin{aligned} M^{g,\eta}(((x_1,\alpha_1),\ldots(x_{r_R},\alpha_{r_R})),R_i) &= L^{\gamma\mathcal{A}}(g)((P^\gamma(x_1,\alpha_1),\ldots,P^\gamma(x_{r_R},\alpha_{r_R})),R) \\ &= L^{\gamma\mathcal{A}}(g)((P^\gamma(y_1,\beta_1),\ldots,P^\gamma(y_{r_T},\beta_{r_T})),R) \\ &= M^{g,\eta}(((y_1,\beta_1),\ldots(y_{r_T},\beta_{r_T})),T). \end{aligned}$$

The first and third equivalences follows from Proposition 8.7. The second equivalence follows from Lemma 8.10 □

We now take quotients of $M^{g,\eta}$ and P^γ with respect to mutual stability. We introduce some notation for these quotients for ease of reading. For each $s \in \boldsymbol{S}$ let $I^{g,\eta}_{\equiv,s} := I^{g,\eta}_s/\equiv$ and let $I^{g,\eta}_\equiv := \uplus_{s\in\boldsymbol{S}} I^{g,\eta}_{\equiv,s}$. For each $\gamma \in [n]^{\underline{A}}$ such that $\gamma^{-1} \sim \eta$ let $P^\gamma_{\equiv,s} : I^{g,\eta}_{\equiv,s} \to X_i$ be defined by $P^\gamma_{\equiv,s}([(x,\alpha)]) := P^\gamma_s(x,\alpha)$ for all $([(x,\alpha)]) \in I^{g,\eta}_{\equiv,s}$. Let $P^\gamma_\equiv := \uplus_{s\in\boldsymbol{S}} P^\gamma_{\equiv,s}$. It follows from Lemma 8.10 that all of these functions are well-defined. Let $M^{g,\eta}_\equiv : \tau[I^{g,\eta}_\equiv] \to \{0,1\}$ be defined by $M^{g,\eta}_\equiv(([(x_1,\alpha_1)],\ldots,[(x_{r_R},\alpha_{r_R})]),R) = M^{g,\eta}(((x_1,\alpha_1),\ldots,(x_{r_R},\alpha_{r_R})),R)$ for each $(([(x_1,\alpha_1)],\ldots,[(x_{r_R},\alpha_{r_R})]),R) \in \tau[I^{g,\eta}_\equiv]$. It follows from Lemma 8.11 that $M^{g,\eta}_\equiv$ is well-defined. We now prove the main result of this section.

Proposition 8.12. *Let* $\gamma \in [n]^{\underline{U}}$ *be such that* $\gamma^{-1} \sim \eta$. *Then* $P^\gamma_\equiv$ *is an isomorphism from* $M^{g,\eta}_\equiv$ *to* $L^{\gamma\mathcal{A}}(g)$.

Proof. It follows from Lemma 8.11 that $\equiv$ is a congruence on $M^{g,\eta}$. It follows from Lemma 8.10 that for all $(x,\alpha),(y,\beta) \in I^{g,\eta}$, $(x,\alpha) \equiv (y,\beta)$ if, and only if, $P^\gamma(x,\alpha) = P^\gamma(y,\beta)$. It follows from Lemma 8.8 that P^γ is an epimorphism from $M^{g,\eta}$ to $L^{\gamma\mathcal{A}}(g)$. We thus have from the first isomorphism theorem that $P^\gamma_\equiv$ is an isomorphism from $M^{g,\eta}_\equiv$ to $L^{\gamma\mathcal{A}}(g)$.

□

8.2 Constructing a Formula

Let $\mathbf{\Omega} = \{\mathbf{\Omega}_1, \ldots, \mathbf{\Omega}_w\}$ be a finite set of P-bounded almost relational vectorised operators. It follows from Lemma 3.16 that we can assume, without a loss of generality, that each of these vectorised operators is Boolean-valued. Let $\mathbb{B} := \mathbb{B}_{\mathbf{\Omega}} \uplus \mathbb{B}_{\mathrm{std}}$. Let $R(\mathbb{B})$ be the set of all relation symbols that appear in the vocabulary of a function in $\mathbb{B}$. Let ρ be a relational vocabulary. Let $\mathcal{C} := (C_n)_{n\in\mathbb{N}}$ be a fixed P-uniform family of transparent symmetric $(\mathbb{B},\rho)$-circuits and let $q \in \mathbb{N}_0$ be such that $\mathcal{C}$ defines a q-ary query. For each $n \in \mathbb{N}$ let $C_n := (G_n, \Omega_n, \Sigma_n, \Lambda_n, L_n)$.

In this section we define a formula $Q \in \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ that defines the same query as $\mathcal{C}$. We now give a brief sketch of the definition of Q. First, we use the Immerman-Vardi theorem to show that there exists a $\mathrm{FP}^{\mathbb{N}}$-interpretation Φ such that for each ρ-structure $\mathcal{A}$ evaluating Φ on $\mathcal{A}$ defines a copy of $C_{|\mathcal{A}|}$ in the number domain. We then define a formula $\theta(\mu,\vec{x})$ such that when g is assigned to μ and denotes a gate and $\vec{a}$ is assigned to $\vec{x}$ and denotes an assignment $\eta_{\vec{a}}$ to the support of g then $\mathcal{A} \models \theta[g,\vec{a}]$ if, and only if, $\eta_{\vec{a}} \in \mathrm{EV}_g$. We define θ by recursion and break the definition into cases. To be more precise, we define a formula $\theta_s(\mu,\vec{x};V)$ for each symbol s denoting either a constant symbol, a relation symbol in ρ, a logical connective, or a vectorised operator in $\mathbf{\Omega}$, such that when g is assigned to μ and denotes a gate associated with s, $\vec{a}$ is assigned to $\vec{x}$ and denotes an assignment $\eta_{\vec{a}}$ to the support of g, and V is a second-order variable assigned to a relation that defines θ for each of the children of g, then $\mathcal{A} \models \theta_s[g,\vec{a}]$ if, and only if, $\eta_{\vec{a}} \in \mathrm{EV}_g$. In other words each θ_s gives a recursive definition of EV_g when g is a gate associated with the symbol s. We define θ by taking a disjunction over these cases to give a recursive definition of EV_g and then using the fixed-point operator to implement recursion and define θ. We finally define Q by using θ to evaluate the output gates of the circuit.

The vast majority of the work of this section is in defining θ_s when $s \in \mathbf{\Omega}$. In this case θ_s gives a recursive definition of EV_g when g is a gate such that $\Sigma_n(g) \in \mathbb{B}_s$. We define θ_s by showing that there is an $\mathrm{FP}^{\mathbb{N}}$-interpretation that defines the structure $M^{g,\eta}_\equiv$ given in Section 8.1. We now work through the technical details and prove Proposition 8.1.

Let $T := \{\mathrm{AND}, \mathrm{OR}, \mathrm{NAND}\} \cup \mathbf{\Omega} \cup \rho \cup \{0,1\}$. It follows from the Immerman-Vardi theorem, the P-uniformity of $\mathcal{C}$, and Lemma 7.4, that there is an $\mathrm{FP}^{\mathbb{N}}[\rho]$-interpretation

$$\Phi := (\phi_G, \phi_\Omega, (\phi_{\Sigma,s})_{s\in T}, (\phi_{\mathbf{\Omega}_i})_{\mathbf{\Omega}_i\in\mathbf{\Omega}}, (\phi_{\Lambda_R})_{R\in\rho}, (\phi_{L,R_i})_{R_i\in R(\mathbb{B})})$$

such that for each $n \in \mathbb{N}$ when Φ is interpreted in ρ-structure $\mathcal{A}$ of size n it defines a symmetric $(\mathbb{B}, \rho)$-circuit with unique labels equivalent to C_n in the number domain. We abuse notation and also refer to this equivalent circuit as C_n. Let t be the width of this interpretation. Throughout this section we use μ, ν, ϵ, η, and δ to denote t-length sequences of number variables and κ and π to denote individual number variables. We now describe the formulas in Φ by describing the relation that each formula defines when interpreted in a ρ-structure $\mathcal{A}$ of size n.

- $\phi_G(\mu)$ defines the set of gates in the circuit C_n as a subset of $[n]^t$. We identify this set with G_n, and hence write $G_n \subseteq [n]^t$.
- $\phi_\Omega(\kappa_1, \ldots, \kappa_q, \mu)$ is defined such that $\mathcal{A} \models \phi_\Omega[a_1, \ldots, a_q, g]$ if, and only if, g is a gate, $(a_1, \ldots, a_q) \in [n]^q$, and $\Omega_n(a_1, \ldots, a_q) = g$.
- $\phi_{\Sigma,s}(\mu)$ is defined for $s \in T$ such that $\mathcal{A} \models \phi_s[g]$ if, and only if, g is an input gate and $\Sigma_n(g) = s$ or g is an internal gate and Σ_n maps g to an element of $\mathcal{B}_s$.
- Let $\mathbf{\Omega}_i \in \mathbf{\Omega}$. Let $s_1, \ldots, s_{l_i}$ be an enumeration of the sort symbols in the vocabulary of $\mathbf{\Omega}_i$ and let $c_1, \ldots, c_{m_i}$ be enumeration of the (constant) function symbols in the vocabulary of $\mathbf{\Omega}_i$. Then $\phi_{\mathbf{\Omega}_i}(\mu, \delta_1, \ldots, \delta_{m_i})$ is such that $\mathcal{A} \models \phi_{\mathbf{\Omega}_i}[g, p_1, \ldots, p_{m_i}]$ if, and only if, g is a gate, $p_1, \ldots, p_{m_i} \in \mathbb{N}_0$, and $\Sigma_n(g) = F_{\mathbf{\Omega}_i,\alpha}[a_1, \ldots, a_{l_i}]$, for some $a_1, \ldots a_{l_i} \in \mathbb{N}$ and where α maps the constant symbols in the vocabulary of $\mathbf{\Omega}_i$ to $\mathbb{N}_0$ such that $\alpha_i(c_j) = p_j$ for each $j \in [m_i]$.
- Let $R \in \rho$. Then $\phi_{\Lambda_R}(\mu, \delta_1, \ldots, \delta_{r_R})$ is such that $\mathcal{A} \models \phi_{\Lambda_R}[g, a_1, \ldots, a_{r_R}]$ if, and only if, $(a_1, \ldots, a_{r_R}) \in [n]^{r_R}$ and g is a relational gate such that $\Sigma(g) = R$ and $(\Lambda_n)_R(g) = (a_1, \ldots, a_{r_R})$.
- Let $R \in R(\mathbb{B})$. Then $\phi_{L,R}(\mu, \nu, \delta_1, \ldots, \delta_{r_R})$ is such that $\mathcal{A} \models \phi_{L,R}[g, h, a_1, \ldots, a_{r_R}]$ if, and only if, g is an internal gate such that R is a relation symbol in the vocabulary of $\Sigma_n(g)$, $h \in H_g$, and $L_n(g)((a_1, \ldots, a_{r_R}), R) = h$.

The interpretation Φ does not define a circuit in exactly the way we might expect. In particular, there is no single formula in Φ that defines Σ_n or L_n for each $n \in \mathbb{N}$. Instead, Φ includes three families of formulas $(\phi_{\Sigma,s})_{s\in T}$, $(\phi_{\mathbf{\Omega}_i})_{\mathbf{\Omega}_i \in \mathbf{\Omega}}$, and $(\phi_{L,R})_{R\in R(\mathbb{B})}$ which together suffice to determine both of these functions. Note as well that ϕ_Ω is used to denote the output gates of the circuit while $\phi_{\mathbf{\Omega}_i}$ is used to denote which vectorised operators (and which assignment to the constants in the vocabulary of those operators) are associated with each gate.

Let n_0 and k be the constants in the statement of Theorem 6.24. Let r' be the maximal arity of a relation in $R(\mathbb{B})$ and let $n_1 = \max(n_0, k \cdot (r' + 1))$. Notice that for each $n \leq n_1$, there are constantly many bijections from the universe of an ρ-structure $\mathcal{A}$ of size n to $[n]$. It follows that here exists a $\mathrm{FP}^{\mathbb{N}}$-formula that evaluates C_n for any $n \leq n_1$ by explicitly quantifying over all of these constantly many bijections, and then evaluating the circuit with

respect to each bijection. For the rest of this section we fix such an $n > n_1$ and let $\mathcal{A}$ denote a ρ-structure of size n.

In the remainder of this section we use μ and ν to denote gates and δ and ϵ to denote elements of the universe of a gate. We use κ, π and λ to denote single number variables and $\vec{\kappa}$ to denote a $2k$-length tuple of number variables. We use $\vec{x}$ and $\vec{y}$ to denote k-length sequences of vertex variables and use $\vec{z}$ to denote $2k$-length sequences of vertex variables. We use U and V to denote second-order variables. If S is a subset of an ordered set we write $\vec{S}$ to denote the $|S|$-tuple given by listing the elements of S in order. Let X and Y be sets, $\vec{a}$ be a sequence in X, and $\vec{u}$ be a sequence of distinct elements in Y such that $|\vec{u}| \leq |\vec{a}|$. Let $\alpha^{\vec{u}}_{\vec{a}} : \mathbf{Img}(\vec{u}) \to X$ be such that $\alpha^{\vec{u}}_{\vec{a}}(b) := \vec{a} \circ \vec{u}^{-1}(b)$ for all $b \in \mathbf{Img}(u)$.

We should like to recursively construct EV_g for each gate g in the circuit. However, while we have from Theorem 6.24 that the canonical support of g has size at most k, it may not be exactly equal to k. If $|\mathrm{sp}(g)| = \ell$, we define

$$\overline{\mathrm{EV}}_g := \{(a_1, \ldots, a_k) \in [n]^k : \alpha^{(a_1,\ldots,a_\ell)}_{\vec{\mathrm{sp}}(g)} \in \mathrm{EV}_g \text{ and } \forall i, j \in [k]\ (i \neq j \implies a_i \neq a_j)\}.$$

We aim to define a $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})[\rho]$-formula $\theta(\mu, \vec{x})$ such that $\mathcal{A} \models \theta[g, \vec{a}]$ if, and only if, $\vec{a} \in \overline{\mathrm{EV}}_g$. We do so by defining for each $s \in T$ a formula θ_s that gives a recursive definition of θ for any gate associated with the symbol s. We now state this formally. Let V be a second-order variable with the same type as $(\mu, \vec{x})$. We aim to define for each $s \in T$ a formula θ_s so that for each gate g with $\mathcal{A} \models \phi_{\Sigma,s}[g]$ and each $\vec{a} \in A^k$, if V is mapped to a relation $\beta(V)$ such that all $h \in H_g$, $(h, \vec{b}) \in \beta(V)$, if, and only if, $\vec{b} \in \overline{\mathrm{EV}}_g$, then $\mathcal{A} \models \theta_s[g, \vec{a}; \beta(V)]$ if, and only if, $\vec{a} \in \overline{\mathrm{EV}}_g$.

Anderson and Dawar [3] have already defined θ_s for each $s \in T \setminus \mathbf{\Omega}$. In each of these cases the definition of θ_s is very straightforward, and so we reproduce these formulas below with minimal discussion and minor adjustments. We first introduce a few auxiliary formulas. We note that the auxiliary formulas SUPP and AGREE have been defined by Anderson and Dawar [3] and we reproduce them here with minor adjustments. We use these formulas to define a number of other auxiliary formulas. We have from the Immerman-Vardi theorem and Lemma 7.8 that there exists a $\mathrm{FP}^{\mathbb{N}}$-formula $\mathrm{SUPP}(\mu, \kappa)$ such that $\mathcal{A} \models \mathrm{SUPP}[g, u]$ if, and only if, g is a gate and $u \in \mathrm{sp}(g)$ [3]. We can define from SUPP a formula SUPP_i for each $i \in \mathbb{N}$ such that $\mathcal{A} \models \mathrm{SUPP}_i[g, u]$ if, and only if, u is the ith element of $\mathrm{sp}(g)$. We define these formulas by induction as follows

$$\begin{aligned}
\mathrm{SUPP}_1(\mu, \kappa) &:\equiv \mathrm{SUPP}(\mu, \kappa) \wedge (\forall \pi(\pi < \kappa) \implies \neg\mathrm{SUPP}(\mu, \pi)) \\
\mathrm{SUPP}_{i+1}(\mu, \kappa) &:\equiv \mathrm{SUPP}(\mu, \kappa) \wedge \exists \pi_1(\pi_1 < \kappa \wedge \mathrm{SUPP}_i(\mu, \pi_1) \\
&\qquad \wedge \forall \pi_2((\pi_1 < \pi_2 < \kappa) \implies \neg\mathrm{SUPP}(\mu, \pi_2))).
\end{aligned}$$

We can define from these formulas a formula $\textsc{agree}(\mu, \nu, \vec{x}, \vec{y})$ such that $\mathcal{A} \models \textsc{agree}(g, h, \vec{a}, \vec{b})$ if, and only if, g and h are gates, $h \in H_g$, and $\alpha^{\vec{a}}_{\vec{\text{sp}}(g)} \sim \alpha^{\vec{b}}_{\vec{\text{sp}}(h)}$ [3]. We define this formula as follows

$$\begin{aligned}\textsc{agree}(\mu, \nu, \vec{x}, \vec{y}) :\equiv \phi_G(\mu) \wedge \phi_G(\nu) \wedge \bigwedge_{1 \leq e,d \leq [k]} [\forall \delta \left(\textsc{supp}_e(\mu, \delta) \wedge \textsc{supp}_d(\nu, \delta)\right) \implies x_e = y_d) \wedge \\ \forall \delta_1, \delta_2 \left((\textsc{supp}_e(\mu, \delta_1) \wedge \textsc{supp}_d(\mu, \delta_2) \wedge x_e = x_d) \implies \delta_1 = \delta_2\right)]\end{aligned}$$

Let $\phi_W(\nu, \mu) :\equiv \bigvee_{R \in R(\mathbb{B})} (\exists \delta_1, \ldots, \delta_{r_R}. \phi_{L,R}(\mu, \nu, \delta_1, \ldots, \delta_{r_R}))$. It can be seen that $\mathcal{A} \models \phi_W[h, g]$ if, and only if, g is an internal gate and $h \in H_g$. We now define for each $s \in T \setminus \boldsymbol{\Omega}$ a formula θ_s as follows

$$\begin{aligned}\theta_0(\mu, \vec{x}) &:\equiv \exists y \, (y \neq y) \\ \theta_1(\mu, \vec{x}) &:\equiv \bigwedge_{1 \leq i < j \leq k} x_i \neq x_j \\ \theta_R(\mu, \vec{x}) &:\equiv (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j) \wedge \exists y_1, \ldots, y_r \exists \kappa_1, \ldots, \kappa_r \, R(y_1, \ldots, y_r) \wedge \phi_{\Lambda_R}(\mu, \kappa_1, \ldots, \kappa_r) \wedge \\ &\quad \bigwedge_{i \in [r]} \bigwedge_{j \in [k]} (\textsc{supp}_j(\mu, \kappa_i) \implies y_i = x_j) \\ \theta_{\text{OR}}(\mu, \vec{x}) &:\equiv (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j) \wedge \exists \nu \exists \vec{y} \, \psi_W(\nu, \mu) \wedge \textsc{agree}(\mu, \nu, \vec{x}, \vec{y}) \wedge V(\nu, \vec{y}) \\ \theta_{\text{AND}}(\mu, \vec{x}) &:\equiv (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j) \wedge \forall \nu \forall \vec{y} \, ((\psi_W(\nu, \mu) \wedge \textsc{agree}(\mu, \nu, \vec{x}, \vec{y})) \implies V(\nu, \vec{y})) \\ \theta_{\text{NAND}}(\mu, \vec{x}) &:\equiv (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j) \wedge \exists \nu \exists \vec{y} \, \psi_W(\nu, \mu) \wedge \textsc{agree}(\mu, \nu, \vec{x}, \vec{y}) \wedge \neg V(\nu, \vec{y})\end{aligned}$$

Let $\boldsymbol{\Omega}_f \in \boldsymbol{\Omega}$. We aim to define $\theta_{\boldsymbol{\Omega}_f}$. Let $\tau := (\boldsymbol{R}, \boldsymbol{F}, \boldsymbol{S}, \zeta)$ be the vocabulary of $\boldsymbol{\Omega}_f$. Let $s_1, \ldots, s_l$ be an enumeration of the sort symbols in $\boldsymbol{S}$ and let $c_1, \ldots, c_m$ be an enumeration of the (constant) symbols in $\boldsymbol{F}$.

We have defined formulas that define the support of a gate and express that two assignments to the supports of two gates are compatible. We now define analogous formulas for the supports of elements of the universe of a gate. Let $s \in \boldsymbol{S}$. From the Immerman-Vardi theorem and Lemma 7.9 there is a formula $\textsc{supp}^s(\mu, \delta, \kappa)$ such that $\mathcal{A} \models \textsc{supp}^s[g, b, u]$ if, and only if, g is a gate, b is an element of s-sort of the universe of g, and $u \in \text{sp}_{\text{sp}(g)}(b)$. We can define for each $i \in \mathbb{N}$, using a similar approach as for SUPP, a formula $\textsc{supp}^s_i(\mu, \delta, \kappa)$ such that $\mathcal{A} \models \textsc{supp}^s_i[g, u, m]$ if, and only if, $\mathcal{A} \models \textsc{supp}^s[g, u, m]$ and m is the ith element of $\text{sp}_{\text{sp}(g)}(u)$. We can use a similar approach as in the definition AGREE to define for each $s \in \boldsymbol{S}$ a formula $\textsc{agree}^s_L(\mu, \nu, \delta, \vec{x}, \vec{y}, \vec{z})$ such that $\mathcal{A} \models \textsc{agree}^s_L[g, h, u, \vec{a}, \vec{b}, \vec{c}]$ if, and only if, g and h are gates with $h \in H_g$, u is an element of the s-sort of the universe of g, and $\alpha^{\vec{a}}_{\vec{\text{sp}}(g)}$, $\alpha^{\vec{b}}_{\vec{\text{sp}}(h)}$, and $\alpha^{\vec{c}}_{\vec{\text{sp}}(u)}$ are all pairwise compatible. Let $\textsc{agree}^s_L(\mu, \delta, \vec{x}, \vec{z}) :\equiv \exists \nu \vec{y} \, \textsc{agree}^s_L(\mu, \nu, \delta, \vec{x}, \vec{y}, \vec{z})$.

We have from Lemma 7.6 and the Immerman-Vardi theorem that for each $s \in \boldsymbol{S}$ there is a formula $\textsc{move}^s(\mu, \delta_1, \delta_2, \vec{\kappa})$ such that $\mathcal{A} \models \textsc{move}^s[g, b_1, b_2, \vec{u}]$ if, and only if, g is a gate, b_1 and b_2 are elements of the s-sort of the universe of g, for all $a, b \in [2k]$ if $a \neq b$ then $u_a \neq u_b$, and there exists $\sigma \in \mathbf{Stab}(\mathrm{sp}(g))$ such that for all $a \in [|\mathrm{sp}(b_1)|]$, $\sigma(\vec{\mathrm{sp}}(b_1)(a)) = u_a$ and $\sigma(b_1) = b_2$. In other words, $\mathcal{A} \models \textsc{move}^s[g, b_1, b_2, \vec{u}]$ if, and only if, the function that maps the support of b_1 to $\vec{u}$ extends to a permutation in $\mathbf{Stab}(\mathrm{sp}(g))$ that maps b_1 to b_2.

For each $s \in \boldsymbol{S}$ let $\textsc{orb}^s(\mu, \delta_1, \delta_2) :\equiv \exists \vec{\kappa}\, \textsc{move}^s(\mu, \delta_1, \delta_2, \vec{\kappa})$. It can be seen that $\mathcal{A} \models \textsc{orb}^s[g, b_1, b_2]$ if, and only if, b_1 and b_2 are elements of the s-sort of the universe of g, and $b_1 \in \mathbf{Orb}(b_2)$. Let

$$\textsc{min-orb}^s(\mu, \delta) :\equiv \forall \epsilon \, (\textsc{orb}^s(\mu, \delta, \epsilon) \implies \delta \leq \epsilon).$$

We use a similar approach as in the definition of $\textsc{supp}_i$ in order to define for each $i \in \mathbb{N}$ and $s \in \boldsymbol{S}$ the formulas $\textsc{out-sp}_i(\mu, \lambda)$ and $\textsc{out-sp}_i^s(\mu, \delta, \lambda)$ such that $\mathcal{A} \models \textsc{out-sp}[g, m]$ if, and only if, g is a gate and m is the ith element of $[n] \setminus \mathrm{sp}(g)$ and $\mathcal{A} \models \textsc{in-sp}_i^s[g, u, m]$ if, and only if, g is an internal gate, u is an element of the s-sort of the universe of g, and m is the ith element of $\mathrm{sp}_{\mathrm{sp}(g)}(u)$.

In Section 8.1 we defined for each gate g and $\eta \in A^{\underline{\mathrm{sp}(g)}}$ a structure $M_{\equiv}^{g,\eta}$. We aim to show that we can define this structure in $\mathrm{FP}^{\mathbb{N}}[\rho]$. We constructed $M_{\equiv}^{g,\eta}$ in stages as follows. First, we defined the functions $\bar{J}^{g,\eta}$ and $J^{g,\eta}$, second, we defined the structure $M^{g,\eta}$, third, we defined the mutual stability relation, and fourth, we defined $M_{\equiv}^{g,\eta}$ by taking a quotient of $M^{g,\eta}$. In order to define $M_{\equiv}^{g,\eta}$ in $\mathrm{FP}^{\mathbb{N}}[\rho]$ we first define formulas corresponding to each of these stages.

Let $R \in \boldsymbol{R}$. We aim to define a formula $\psi_{\bar{J},R}(\mu, \vec{x}, \delta_1, \vec{z}_1, \ldots, \delta_{r_R}, \vec{z}_{r_R}, \vec{\kappa}_1, \ldots, \vec{\kappa}_{r_R})$ such that $\mathcal{A} \models \psi_{\bar{J},R}[g, \vec{a}, u_1, \vec{c}_1, \ldots, u_{r_R}, \vec{c}_{r_R}, \vec{m}_1, \ldots, \vec{m}_{r_R}]$ if, and only if, g is a gate, for each $j \in [r_R]$ we have that u_i is element of the $\zeta(R)(i)$-sort of the universe of g, and

$$\bar{J}^{g,\alpha^{\vec{a}}_{\vec{\mathrm{sp}}(g)}}((u_1, \alpha^{\vec{c}_1}_{\vec{\mathrm{sp}}(u_1)}), \ldots, (u_{r_R}, \alpha^{\vec{c}_{r_R}}_{\vec{\mathrm{sp}}(u_{r_R})})) = (\alpha^{\vec{m}_1}_{\vec{\mathrm{sp}}(u_1)}, \ldots, \alpha^{\vec{m}_{r_R}}_{\vec{\mathrm{sp}}(u_{r_R})}).$$

We define this formula recursively. We first define a set of auxiliary formulas. For each $p \in \mathbb{N}$ and $j \in [2k]$ let

$$\begin{aligned}\psi^j_{R,p}(\mu, \vec{x}, \delta, \vec{z}, \vec{\kappa}) :\equiv &(\forall \lambda \, \neg \textsc{supp}_j^{\zeta(R)(p)}(\mu, \delta, \lambda)) \vee \exists \lambda \, [\textsc{supp}_j^{\zeta(R)(p)}(\mu, \delta, \lambda) \wedge [(\textsc{supp}(\mu, \lambda) \wedge \vec{\kappa}(j) = \lambda) \vee \\ &(\neg \textsc{supp}(\mu, \lambda) \wedge \bigvee_{a \in [2k]} \textsc{in-sp}_a^{\zeta(R)(p)}(\mu, \delta, \lambda) \wedge \textsc{out-sp}_{(p-1)k+a}(\mu, \vec{\kappa}(j)))]].\end{aligned}$$

This formula will be used for both the base and inductive cases in the definition of $\bar{J}^{g,\eta}$. The sequence $\vec{\kappa}$ is intended to encode a function that maps the support of δ to $\vec{\kappa}$. The purpose of this formula is to check if there is some element u that is the jth element of the support of δ and, if there is, to check that (i) if u is in the support of μ then the function encoded

by $\vec{\kappa}$ fixes u, and (ii) if u is not in the support of μ then the function encoded by $\vec{\kappa}$ maps u to an appropriate element outside the support of μ. We now define for each $p \in [r_R]$ a $\mathrm{FP}^{\mathbb{N}}$-formula $\psi^p_{\bar{J},R}$ as follows. Let

$$\psi^1_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\vec{\kappa}_1) :\equiv \bigwedge_{1\le a<b\le k}(x_a \neq x_b) \wedge \bigwedge_{1\le a<b\le 2k}(z_a \neq z_b \wedge \kappa_a \neq \kappa_b) \wedge \\ \textsc{agree}_L^{\zeta(R)(p)}(\mu,\delta,\vec{x},\vec{z}) \wedge \bigwedge_{j\in[2k]} \psi^j_{R,1}(\mu,\vec{x},\delta_1,\vec{z}_1,\vec{\kappa}_1),$$

and let

$$\psi^{p+1}_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{p+1},\vec{z}_{p+1},\vec{\kappa}_1,\ldots,\vec{\kappa}_{p+1}) :\equiv \psi^p_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_p,\vec{z}_p,\vec{\kappa}_1,\ldots,\vec{\kappa}_p) \wedge$$

$$\bigwedge_{1\le a<b\le 2k}(\vec{z}_{p+1}(a) \neq \vec{z}_{p+1}(b) \wedge \vec{\kappa}_{p+1}(a) \neq \vec{\kappa}_{p+1}(b)) \wedge \textsc{agree}_L^{\zeta(R)(p+1)}(\mu,\delta_{p+1},\vec{x},\vec{z}_{p+1}) \wedge$$

$$\bigwedge_{b\in[2k]}[(\exists\lambda\ \textsc{supp}_b^{\zeta(R)(p+1)}(g,\delta_{p+1},\lambda)) \implies [[(\bigwedge_{a\in[p]}\bigwedge_{d\in[2k]}[(\exists\lambda_1 \textsc{supp}_d^{\zeta(R)(j)}(g,\delta_a,\lambda_1)) \implies$$

$$\vec{z}_{p+1}(b) \neq \vec{z}_a(d)]) \wedge \psi^b_{R,p+1}(\mu,\vec{x},\delta_{p+1},\vec{z}_{p+1},\vec{\kappa}_{p+1})] \vee$$

$$[\bigvee_{a\in[p]}\bigvee_{d\in[2k]}(\exists\lambda_1 \textsc{supp}_d^{\zeta(R)(j)}(g,\delta_a,\lambda_1))) \wedge \vec{z}_{p+1}(b) = \vec{z}_a(d) \wedge \vec{\kappa}_{p+1}(b) = \vec{\kappa}_a(d))]]]$$

The purpose of the second line of this formula is to check that the assignment to the support of δ_{p+1} and the mapping given by $\vec{\kappa}_{p+1}$ are injections and that the assignments to the supports of μ and δ_{p+1} are compatible. The rest of the formula is intended to handle the three cases that appear in the definition of $\bar{J}^{g,\eta}$. More formally, the purpose of the third, fourth, and fifth lines is to check that for every $b \in [2k]$ if there is some u that is the bth element of the support of δ_{p+1} then either (i) for every $a \in [p]$ we have $\vec{z}_a(b) \notin \vec{z}_{p+1}$ and $\vec{\kappa}_{p+1}$ fixes u if u is in the support of μ and otherwise moves u to an appropriate point outside the support of μ, or (ii) there exists some $a \in [p]$ and $d \in [2k]$ such that $\vec{z}_{p+1}(b) = \vec{z}_a(d)$, and the permutation encoded by $\vec{\kappa}_{p+1}$ maps u to the same element that $\vec{\kappa}_a$ maps the dth element of the support of δ_{p+1}. We let

$$\psi_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{r_R},\vec{z}_{r_R},\vec{\kappa}_1,\ldots,\vec{\kappa}_{r_R}) :\equiv \psi^{r_R}_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{r_R},\vec{z}_{r_R},\vec{\kappa}_1,\ldots,\vec{\kappa}_{r_R}).$$

We now define $\psi_{J,R}(\mu,\vec{x},\delta_1\vec{z}_1,\ldots,\delta_{r_R}\vec{z}_{r_R},\nu,\vec{y})$ such that $\mathcal{A} \models \psi_{J,R}[g,\vec{a},u_1,\vec{c}_1,\ldots,u_{r_i},\vec{c}_{r_R},h,\vec{b}]$ if, and only if, g is an internal gate, $h \in H_g$, and $J^{g,\alpha^{\vec{a}}_{\mathrm{sp}(g)}}((u_1,\alpha^{\vec{c}_1}_{\mathrm{sp}(u_1)}),\ldots,(u_{r_R},\alpha^{\vec{c}_{r_R}}_{\mathrm{sp}(u_{r_R})})) =$

$(h, \alpha^{\vec{b}}_{\vec{\mathrm{sp}}(h)})$. This formula is defined as follows

$$\begin{aligned}\psi_{J,R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,&\delta_{r_R},\vec{z}_{r_R},\nu,\vec{y}) :\equiv \exists\vec{\kappa}_1,\ldots,\vec{\kappa}_{r_R}[\psi_{\bar{J},R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{r_R},\vec{z}_{r_R},\vec{\kappa}_1,\ldots,\vec{\kappa}_{r_i})\wedge\\
&\exists\delta'_1,\ldots,\delta'_{r_R}(\phi_{L,R_i}(\mu,\nu,\delta'_1,\ldots,\delta'_{r_R})\wedge\\
&\exists\vec{z}'_1,\ldots,\vec{z}'_{r_R}[\bigwedge_{j\in[r_R]}[\textsc{move}^{\zeta(R)(j)}(g,\delta_j,\delta'_j,\vec{\kappa}_j)\wedge\textsc{agree}_L^{\zeta(R)(j)}(\mu,\nu,\delta'_j,\vec{x},\vec{y},\vec{z}'_j)\wedge\\
&[\bigwedge_{1\le a<b\le 2k}\vec{z}'_j(a)\neq\vec{z}'_j(b)\wedge\bigwedge_{a\in[2k]}(\forall\lambda\,(\neg\textsc{supp}^{\zeta(R)(j)}(\mu,\delta',\lambda)))\vee\\
&\bigvee_{b\in[2k]}\textsc{supp}_a^{\zeta(R)(j)}(\mu,\delta',\vec{\kappa}_j(b))\wedge\vec{z}'_j(a)=\vec{z}_j(b))]]]\end{aligned}$$

The purpose of the first line is to define the functions $\vec{\kappa}_1,\ldots,\vec{\kappa}_{r_R}$ as per the definition of $\psi_{\bar{J},R}$. The purpose of the second line and the first part of the third line is to define δ'_j for each $j\in[r_R]$ such that δ'_j is the image of δ_j under the action of any permutation extending the function $\vec{\kappa}_j$. For each $j\in[r_R]$ we have an assignment to the support of δ'_j given by mapping $\vec{\kappa}_j$ to $\vec{z}_j$. From Lemma 8.4 this set of assignments is pairwise compatible. Notice that $\vec{\kappa}_j$ contains the support of δ'_j, but perhaps not in order. The purpose of the second part of the third line and lines four and five is to re-order each $\vec{z}_j$ so as to match the order on the support of δ'_j, and then to check if the assignment $\vec{y}$ to ν is compatible with these assignments.

We now define $\psi_{M,R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{r_R},z_{r_R};V)$ such that $\mathcal{A}\models\psi_{M,R}[g,\vec{a},u_1,\vec{c}_1,\ldots u_{r_R},\vec{c}_{r_R};\beta(V)]$ if, and only if, $M^{g,\alpha^{\vec{a}}_{\vec{\mathrm{sp}}(g)}}(((u_1,\alpha^{\vec{c}_1}_{\vec{\mathrm{sp}}(u_1)}),\ldots,(u_{r_R},\alpha^{\vec{c}_{r_R}}_{\vec{\mathrm{sp}}(u_{r_i})})),R)=1$, where $\beta(V)$ is an assignment to V such that for all $h\in H_g$ and $\vec{b}\in A^k$, $(h,\vec{b})\in\beta(V)$ if, and only if, $\vec{b}\in\overline{\mathrm{EV}}_h$. This formula is defined as follows

$$\psi_{M,R}(\mu,\vec{x},\delta_1,\vec{z}_1,\ldots,\delta_{r_R},\vec{z}_{r_R};V) :\equiv \exists\nu,\vec{y}\,(\psi_{J,R}(\mu,\vec{z},\delta_1,\vec{z}_1,\ldots,\delta_{r_R}\vec{z}_{r_R},\nu,\vec{y})\wedge V(\nu,\vec{y})).$$

We now define a formula $\psi^D_j(\mu,\vec{x},\delta,\vec{z})$ for each $s\in\boldsymbol{S}$ such that $\mathcal{A}\models\psi^D_s[g,\vec{a},u,\vec{c}]$ if, and only if, $(u,\alpha^{\vec{c}}_{\vec{\mathrm{sp}}(u)})\in I_j^{g,\alpha^{\vec{a}}_{\vec{\mathrm{sp}}(g)}}$. For each $s\in\boldsymbol{S}$ we define this formula as follows

$$\psi^D_s(\mu,\vec{x},\delta,\vec{z}) :\equiv \textsc{min-orbit}^s(\mu,\delta)\wedge\textsc{agree}^s_L(\mu,\delta,\vec{x},\vec{z}).$$

We now define a formula $\psi^{\approx}_s(\mu,\vec{x},\delta_1,\vec{z}_1,\delta_2,\vec{z}_2)$ for each $s\in\boldsymbol{S}$ such that $\mathcal{A}\models\psi^{\approx}_s[g,\vec{a},u_1,\vec{c}_1,u_2,\vec{c}_2]$ if, and only if, g is a gate, u_1 and u_2 are elements of the s-sort of the universe of g, and

$(u_1, \alpha^{\vec{c}_1}_{\vec{\mathrm{sp}}(u_1)}) \equiv (u_2, \alpha^{\vec{c}_2}_{\vec{\mathrm{sp}}(u_2)})$. For each $s \in \boldsymbol{S}$ we define this formula as follows

$$\begin{aligned}\psi^{\cong}_s(\mu, \vec{x}, \delta_1, \vec{z}_1, \delta_2, \vec{z}_2) :\equiv & \bigwedge_{1 \leq a < b \leq k} (x_a \neq x_b) \wedge \bigwedge_{1 \leq a < b \leq 2k} (\vec{z}_1(a) \neq \vec{z}_1(b) \wedge \vec{z}_2(a) \neq \vec{z}_2(b)) \wedge \\ & \textsc{agree}^s_L(\mu, \delta_1, \vec{x}, \vec{z}_1) \wedge \textsc{agree}^s_L(\mu, \delta_2, \vec{x}, \vec{z}_2) \wedge \delta_1 = \delta_2 \wedge \\ & \exists \vec{\kappa} \, [(\bigwedge_{1 \leq a < b \leq 2k} \kappa_a \neq \kappa_b) \wedge \textsc{move}^s(\mu, \delta_1, \delta_1, \vec{\kappa}) \wedge \\ & \bigwedge_{a \in [2k]} [(\exists \lambda \textsc{supp}^s_a(\mu, \delta_1, \lambda) \implies [(\textsc{supp}(\mu, \kappa_a) \implies \textsc{supp}^s_a(\mu, \delta_1, \kappa_a)) \wedge \\ & \bigvee_{b \in [2k]} \textsc{supp}^s_b(\mu, \delta_1, \kappa_a) \wedge \vec{z}_1(a) = \vec{z}_2(b))]]]\end{aligned}$$

The purpose of the first line is to check that the assignments to the supports of μ, δ_1, and δ_2 are injections. The purpose of the second line is to check that the assignments to δ_1 and δ_2 are compatible with the assignment to μ and that $\delta_1 = \delta_2$. The purpose of the third line is check if there exists a permutation σ extending the function encoded by $\vec{\kappa}$ that fixes δ_1. The purpose of the fourth and fifth line is to check that σ also fixes those elements in the support of μ, fixes the support of δ_1 setwise, and that applying σ maps the assignments $\vec{z}_1$ and $\vec{z}_2$ appropriately. We define $\theta_{\boldsymbol{\Omega}_f}$ as follows

$$\begin{aligned}\theta_{\boldsymbol{\Omega}_f}(\mu, \vec{x}; V) :\equiv & \, (\bigwedge_{1 \leq i < j \leq k} x_i \neq x_j) \wedge \exists \epsilon_1, \ldots, \epsilon_m \, (\phi_{\boldsymbol{\Omega}_f}(\mu, \epsilon_1, \ldots, \epsilon_m) \wedge \\ & \boldsymbol{\Omega}_f[(\psi^D_s)_{s \in \boldsymbol{S}}, (\psi^{\cong}_s)_{s \in \boldsymbol{S}}][(\vec{y}_1 \delta_1, \ldots, \vec{y}_{r_R} \delta_{r_R}) \psi_{M,R}(\mu, \vec{x}, \delta_1, \vec{y}_1, \ldots, \delta_{r_R}, \vec{y}_{r_R}; V)]_{R \in \boldsymbol{R}}[\epsilon_i]_{c_i \in \boldsymbol{F}}).\end{aligned}$$

We now show that $\theta_{\boldsymbol{\Omega}_f}$ suffices for this recursive definition.

Lemma 8.13. *Let g be a gate such that $\Sigma_n(g) \in \mathbb{B}_{\boldsymbol{\Omega}_f}$ and let $\vec{a} \in A^{\underline{k}}$. Let β be an assignment that maps μ to g, $\vec{x}$ to $\vec{a}$, and V to $\beta(V)$ such that for all $h \in H_g$ and all $\vec{b} \in A^{\underline{k}}$ we have $(h, \vec{b}) \in \beta(V)$ if, and only if , $\vec{b} \in \overline{EV}_h$. Then $\mathcal{A} \models \theta_{\boldsymbol{\Omega}_f}[\beta]$ if, and only if, $\vec{a} \in \overline{EV}_g$.*

Proof. Let $\eta := \alpha^{\vec{a}}_{\vec{\mathrm{sp}}(g)}$. Let $\tau_{\mathbf{rel}} = (\boldsymbol{R}, \boldsymbol{S}, \zeta|_{\boldsymbol{R}})$. Let $\mathcal{I} := \langle (\psi^D_s)_{s \in \boldsymbol{S}}, (\psi^{\cong}_s)_{s \in \boldsymbol{S}}, (\psi_{M,R})_{R \in \boldsymbol{R}} \rangle$. Let $p_1, \ldots, p_m$ be such that $\mathcal{A} \models \phi_{\boldsymbol{\Omega}_f}[\beta \frac{p_1}{\pi_1} \ldots \frac{p_m}{\pi_m}]$. Let $\mathcal{B}$ be the quotient of the $\tau_{\mathbf{rel}}$-structure $(\uplus_{s \in \boldsymbol{S}} (\psi^D_s)^{(\mathcal{A}, \beta)}, (\psi^{(\mathcal{A}, \beta)}_{M,R})_{R \in \boldsymbol{R}})$ by the equivalence relation $\approx := \uplus_{s \in \boldsymbol{S}} (\psi^{\cong}_s)^{(\mathcal{A}, \beta)}$. It follows that $\mathcal{B} = \mathcal{I}(\mathcal{A}, \beta)$. For each $s \in \boldsymbol{S}$ let $K_s : I^{g,\eta}_{\equiv,s} \to B_s$ be defined for all $[(d, \alpha)] \in I^{g,\eta}_{\equiv,s}$ by $K_s([(d, \alpha)]) := [(d, \vec{e})]$, where $\vec{e} \in A^{\underline{2k}}$ such that $\alpha(\vec{e}(i)) = \vec{\mathrm{sp}}(a)(i)$ for all $i \in [|\mathrm{sp}(d)|]$. Let $K = \uplus_{s \in \boldsymbol{S}} K_s$. It can be shown that K is well-defined and bijective. Let $R \in \boldsymbol{R}$ and let $(([(d_1, \alpha_1)], \ldots, [(d_{r_R}, \alpha_{r_R})]), R) \in I^{g,\eta}_{\equiv}$. Then $M^{g,\eta}_{\equiv}(([(d_1, \alpha_1)], \ldots, [(d_{r_R}, \alpha_{r_R})]), R) = 1$ if, and only if, $\mathcal{A} \models \psi_{M,R}[g, \vec{a}, d_1, \vec{e}_1, \ldots, d_{r_R}, \vec{e}_{r_R}; \beta(V)]$, where for all $j \in [r_R]$, $[(d_j, \vec{e}_j)] = K([(d_j, \alpha_j)])$ if, and only if, $(K([d_1, \alpha_1]), \ldots, K([(d_{r_R}, \alpha_{r_R})])) \in R^{\mathcal{B}}$. It follows that K is an isomorphism.

Let $\alpha : \boldsymbol{F} \to \mathbb{N}_0$ be defined such that $\alpha(c_i) = p_i$ for each $c_i \in \boldsymbol{F}$. Let $\mathcal{B}^*$ be a τ-structure such that $\mathcal{B}^*$ is an expansion of $\mathcal{B}$ with $c^{\mathcal{B}^*} = p_i$ for each $c_i \in \boldsymbol{F}$. Let M^* be a τ-structure such that M^* is an expansion of $M^{g,\eta}_{\equiv}$ with $c_i^{M^*} = p_i$ for each $c_i \in \boldsymbol{F}$. We have that $\mathcal{B}^*$ and M^* are isomorphic. Let E be the evaluation function of $\boldsymbol{\Omega}_f$ and let $\gamma \in [n]^{\underline{A}}$ be such that $\gamma^{-1} \sim \eta$. Then

$$\begin{aligned}\mathcal{A} \models \theta_{\boldsymbol{\Omega}_f}[\beta] &\iff E(\mathcal{B}^*) = 1 \iff E(M^*) = 1 \iff F_{\boldsymbol{\Omega}_f,\alpha}[I^{g,\eta}_{\equiv}](M^{g,\eta}_{\equiv}) = 1 \\ &\iff \Sigma_n(L^{\gamma\mathcal{A}}(g)) = 1 \iff \eta \in \mathrm{EV}_g \iff \vec{a} \in \overline{\mathrm{EV}}_g\end{aligned}$$

The second equivalence follows from the fact that $\mathcal{B}$ and $M^{g,\eta}_{\equiv}$ are isomorphic and the fourth equivalence follows Proposition 8.12. □

We now define $\theta(\mu, \vec{x})$ as follows

$$\theta(\mu, \vec{x}) :\equiv [\mathbf{ifp}_{V,\nu\vec{y}} \bigvee_{s\in T} (\phi_s(\mu) \wedge \theta_s(\nu, \vec{y}))](\mu, \vec{x}).$$

Let g be a gate and $\vec{a} \in A^{\underline{k}}$. Let β be an assignments to V such that for all $h \in H_g$ and all $\vec{b} \in A^{\underline{k}}$ we have $(h, \vec{b}) \in \beta(V)$ if, and only if , $\vec{b} \in \overline{\mathrm{EV}}_h$. It is easy to show that for each $s \in T \setminus \boldsymbol{\Omega}$, $\mathcal{A} \models \theta_s[g, \vec{a}; \beta(V)]$ if, and only if, $\vec{a} \in \overline{\mathrm{EV}}_g$. It thus follows from Lemma 8.13 that for all $s \in T$, $\theta_s[g, \vec{a}; \beta(V)]$ if, and only if, $\vec{a} \in \overline{\mathrm{EV}}_g$. From this it can be shown, using a straight-forward inductive argument, that for all $g \in G$ and $\vec{a} \in A^{\underline{k}}$, $\mathcal{A} \models \theta[g, \vec{a}]$ if, and only if, $\vec{a} \in \overline{\mathrm{EV}}_g$.

We now define a $\mathrm{FP}^{\mathbb{N}}(\boldsymbol{\Omega})[\rho]$-formula Q that defines the same q-ary query as $\mathcal{C}$. The definition of this formula is similar to one given in [3]. Let

$$\begin{aligned}Q(y_1, \ldots y_q) :\equiv \exists\vec{x}\, \exists\mu, \kappa_1, \ldots, \kappa_q \pi_1, \ldots, \pi_k \,[&\theta(\mu, \vec{x}) \wedge \phi_\Omega(\kappa_1, \ldots, \kappa_q, \mu) \wedge \\ &\bigwedge_{1\le i\le k} (\mathrm{SUPP}_i(\mu, \pi_i) \vee \forall\pi\, (\neg\mathrm{SUPP}_i(\mu, \pi))) \wedge \\ &\bigwedge_{1\le i\le k} \bigwedge_{1\le j\le q} ((\mathrm{SUPP}_i(\mu, \pi_i) \wedge (x_i = y_j)) \implies \kappa_j = \pi_i) \wedge \\ &\bigwedge_{1\le j\le q} \bigvee_{1\le i\le k} (x_i = y_j \wedge \mathrm{SUPP}_i(\mu, \pi_i))].\end{aligned}$$

We could informally understand the formula Q as inverting the assignment denoted by $(y_1, \ldots, y_q)$, selecting the corresponding output gate, and then evaluating this output gate. This completes the proof of Proposition 8.1.

Chapter 9

The Main Result

In this brief chapter we restate and prove the main theorem. We also discuss a number of noteworthy corollaries of this theorem.

Theorem 4.19 (Main Theorem)**.** Let $\mathbf{\Omega}$ be a finite union of P-bounded almost relational vectorised operators. Let ρ be a relational vocabulary. Then

1. Every query definable in $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$ is definable by a P-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuits, and
2. Every query definable by a P-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}, \rho)$-circuits is definable in $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$.

Proof. The first claim follows from Theorem 5.5 and the second claim follows from Proposition 8.1. □

If an extension of fixed-point logic is closed under operator quotients then we have an exact circuit characterisation of the logic. We state this observation formally.

Corollary 9.1. *Let $\mathbf{\Omega}$ be a finite union of* P*-bounded almost relational vectorised operators. Let ρ be a relational vocabulary. If* $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ *is closed under operator quotients then a query is definable in* $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ *if, and only if, it is definable by a* P*-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{std}, \rho)$-circuits.*

We notice that by setting $\mathbf{\Omega}$ to be either the empty set or the counting operator we can recover the symmetric circuit characterisations of $\mathrm{FP}^{\mathbb{N}}$ and FPC established by Anderson and Dawar [3].

We have established circuit characterisations for a particular class of logics. It is a straightforward consequence of a result by Dawar [11] that with respect to the search for a logic for P we can restrict our attention to logics of this form. We state this observation formally. We say that a vectorised (many-sorted) quantifier is *polynomial-time decidable* if the corresponding class of structures is polynomial-time decidable.

Proposition 9.2. *There is a logic that captures polynomial-time if, and only if, there exists a polynomial-time decidable vectorised many-sorted quantifier* $\mathbf{\Omega}$ *such that* $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ *is closed under operator quotients and captures polynomial-time.*

Proof. Suppose there exists a logic that captures polynomial-time. It follows from Theorem 2.9 that exists a polynomial-time decidable vectorised quantifier Q such that $\mathrm{FO}(\widetilde{Q})$ captures polynomial-time. Let $\mathbf{\Omega}$ be the definition of Q as a vectorised quantifier. It is easy to see that any formula in $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ must decide a query in P. It follows that $\mathrm{FO}(\widetilde{Q}) \leq \mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}) \leq \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega}) \leq \mathrm{FO}(\widetilde{Q})$ and so $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}}) \equiv \mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ and $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ captures polynomial-time. □

We can use Theorem 4.19 to state the question of whether there is a logic for P purely in terms of symmetric circuits. We formalise this now.

Definition 9.3. We call a basis $\mathbb{B}$ *polynomial-time bounded* if there exists a polynomial p and computable function that maps each function $F \in \mathbb{B}$ to a Turing machine M_F that computes F and such that $M_F(x)$ has running time bounded by $p(|x|)$ for x an input string.

Proposition 9.4. *There is a logic that captures polynomial-time if, and only if, there is a finitely generated polynomial-time bounded basis* $\mathbb{B}$ *such that a query is decidable in polynomial-time if, and only if, it is definable by a* P*-uniform family of transparent symmetric circuits defined over* $\mathbb{B}$*.*

Proof. '⇒': It follows from Proposition 9.2 that there is a polynomial-time computable vectorised many-sorted quantifier $\mathbf{\Omega}$ such that $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ is closed under operator quotients and captures polynomial-time. It follows from the fact that $\mathbf{\Omega}$ is polynomial-time computable that $\mathbb{B}_{\mathbf{\Omega}}$ is finitely generated and polynomial-time bounded. It follows from Theorem 4.19 for any query Q that Q is definable in polynomial-time if, and only if, Q is definable in $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ if, and only if, Q is definable by a P-uniform family of transparent symmetric circuits defined over $\mathbb{B}_{\mathbf{\Omega}}$.

'⇐': Let $\mathbb{B}$ be a finitely generated polynomial-time bounded basis satisfying the hypothesis. Since $\mathbb{B}$ is finitely generated there exists classes of structures $\mathcal{G}_1, \ldots, \mathcal{G}_k$ each of which is polynomial-time decidable and such that $\mathbb{B} = \bigcup_{i \in [k]} \mathbb{B}_{\mathcal{G}_i}$. For each $i \in [k]$ let Q_i be the vectorised quantifier generated by $\mathcal{G}_i$. Let $\mathbf{Q} = \{Q_1, \ldots, Q_k\}$. Then $\mathrm{FP}^{\mathbb{N}}(\mathbf{Q})$ captures polynomial-time. □

Theorem 4.19 characterises families of *transparent* symmetric circuits in terms of fixed-point logics. The restriction to transparent circuits is important as it ensures the polynomial-time decidability of a number of circuit parameters, and we need this to define the translation from families of circuits to formulas. However, unless P-uniform families of symmetric circuits and transparent circuits have the same expressive power, this leaves us without a characterisation for general symmetric circuits. We now show that if there is a basis for which

P-uniform families of symmetric circuits and transparent circuits have different expressive powers then the graph isomorphism problem is not in P.

Proposition 9.5. *Let $\mathbb{B}$ be a basis and ρ be a relational vocabulary. If the graph isomorphism problem is in* P *then every query decidable by a* P*-uniform family of symmetric $(\mathbb{B}, \rho)$-circuits is definable by any* P*-uniform family of transparent symmetric $(\mathbb{B}, \rho)$-circuits.*

Proof. Suppose graph-isomorphism is polynomial-time decidable. Then, using a similar approach as in Lemma 7.1, we can define an algorithm that runs in polynomial time and takes as input a symmetric circuit and outputs the syntactic-equivalence classes of the gates in that circuit. We can then show, using the same approach as in Lemme 7.4, that there is a polynomial-time computable mapping that takes as input a symmetric circuit C and outputs a reduced injective symmetric circuit C' such that C and C' compute the same function. It follows that if Q is a query definable by a P-uniform family of symmetric $(\mathbb{B}, \rho)$-circuits $(C_n)_{n\in\mathbb{N}}$ then there exists a P-uniform family of reduced injective symmetric $(\mathbb{B}, \rho)$-circuits $(C'_n)_{n\in\mathbb{N}}$ that also define the query Q. We recall that a reduced injective circuit is transparent, and so the result follows.

□

We conclude this chapter by discussing the particular case of FPR. The study of the expressive power of FPR and the question of whether it captures polynomial-time remain topics of particular interest in descriptive complexity. The work of this thesis was, in part, motivated by the desire to develop a circuit characterisation for FPR. This characterisation follows from the main theorem and, because of the particular importance of FPR, we now work through the details here. We first define the rank basis explicitly.

Definition 9.6. Let $p, t \in \mathbb{N}$ and suppose p is prime. Let $\tau := (\{R\}, [3], \zeta)$. Let $\zeta(R) = (1, 2, 3)$. For each $a, b, c \in \mathbb{N}$ let $\mathbf{rk}_p^t[a, b, c] : \{0,1\}^{\tau[a,b,c]} \to \{0,1\}$ be defined for each $\mathcal{B} \in \{0,1\}^{\tau[a,b,c]}$ as follows. Let $M_\mathcal{B} : [a] \times [b] \to \mathbb{F}_p$ be defined such that

$$M_\mathcal{B}(x, y) := |\{z \in [c] : (x, y, z) \in R^\mathcal{B}\}| \mod p$$

for all $(x, y) \in [a] \times [b]$. Let $\mathbf{rk}_p^t(\mathcal{B}) = 1$ if, and only if, the rank of $\mathbf{rk}(M_\mathcal{B}) \geq t$. Let $\mathbb{B}_{\mathbf{rk}}$ be the set of all such Boolean functions for any $p, t, a, b, c \in \mathbb{N}$ with p prime. We call $\mathbb{B}_{\mathbf{rk}}$ the *rank basis.*

It is easy to see that the rank basis $\mathbb{B}_{\mathbf{rk}}$ is precisely the basis corresponding the rank operator $\mathbf{\Omega}_{E_{\mathbf{rk}}}$ defined in Section 3.2. We call a circuit defined over the union of the standard basis and the rank basis a *rank circuit.* From Theorem 4.19 we have the following circuit characterisation of FPR.

Theorem 9.7. *A query is definable in* FPR *if, and only if, it is definable by a* P*-uniform family of transparent symmetric rank circuits.*

Proof. It follows from Lemma 3.5 that FPR is closed under operator quotients. This result then follows immediately from Theorem 4.19. □

Chapter 10

Conclusions and Future Work

In this chapter we summarise the main findings of this thesis and discuss some interesting areas for possible future study.

10.1 Summary and Discussion

The study of fixed-point logics and their extensions is of central importance in finite model theory and, in particular, in the search for a logic for polynomial-time. In this thesis we have shown that each from a broad class of extensions of fixed-point logic can be characterised by uniform families of transparent symmetric circuits over a corresponding basis. In order to prove this result we developed new more general frameworks both for studying extensions of logics and for studying circuits defined over a broader class of bases.

This result establishes a deep and interesting connection between circuit complexity and descriptive complexity. Most immediately, it allows us to translate inexpressibility results for the logic to lower-bounds for classes of symmetric circuits, and vise versa. This gives us the opportunity to exploit combinations of techniques from both logic and circuit complexity in order to solve problems in both fields.

It also follows from the main result that we have a circuit characterisation of FPR. We recall that FPR is one of the strongest logics known to be in polynomial-time but not known to capture it. For this reason precisely characterising the expressive power of FPR is an active area of research. The circuit characterisation established here provides us with an alternative characterisation of the expressive power of FPR, opening up new avenues for proving lower-bounds.

Perhaps most interestingly, the main result of this thesis presents an interesting perspective on the central tension in descriptive complexity, the relationship between machine models of computation and model-theoretic logics. We should explain this in more detail. We recall that an algorithm specified by a machine model can be translated to a uniform family of circuits. In contrast, each formula in a logic can be translated to a uniform family of *symmetric*

circuits. In this sense the formulas of a logic each define an algorithm that uses only those techniques that respect the inherent symmetries of the structure. The main theorem of this thesis establishes that fixed-point logics do not just define uniform families of symmetric circuits (over an appropriate basis), but rather they are characterised by these P-uniform families of symmetric circuits. In this sense we can understand symmetry as the defining feature of a logic, and our framework provides the tools for studying it. To illustrate this point we consider a specific case. Let $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ be a logic closed under operator-quotients, where $\mathbf{\Omega}$ is a finite union of polynomial-time computable P-bounded almost relational vectorised operators. We recall that a query is polynomial-time decidable if, and only if, it is decidable by P-uniform families of *invariant* circuits over $\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}$. In contrast, the logic $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ can define exactly those queries definable by P-uniform families of transparent *symmetric* circuits over the basis $\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}$. Here we notice the fundamental role symmetry plays as exactly that property separating the logic from the machine model. The question of whether they have the same expressive power (i.e. whether $\mathrm{FP}^{\mathbb{N}}(\mathbf{\Omega})$ captures polynomial-time) is equivalent to the question of whether P-uniform families of symmetric and invariant circuits over $\mathbb{B}_{\mathbf{\Omega}} \cup \mathbb{B}_{\mathrm{std}}$ have the same expressive power. In this way, the question of whether there is a logic for P, the central question in the field, can be understood as essentially a question about whether there is a strong enough basis over which polynomial-time computation is in this sense 'inherently symmetric'.

We now review some of the important contributions of this thesis and discuss some of the novel frameworks and techniques developed.

We began by defining a general framework for studying extensions of fixed-point logic. We introduced the notion of a generalised operator. We showed that Lindström quantifiers, counting operators, and rank operators can all be defined in this framework. We showed that each family of P-bounded almost relational generalised operators $\mathbf{\Omega}$ can be associated with a family of quantifiers $\mathbf{Q}_{\mathbf{\Omega}}$. This generalises the relationship between counting operators and counting quantifiers. We aimed to generalise the translation from FPC to P-uniform families of FO+C-formulas. However, we noted that the usual translation given by unrolling fixed-points does, in general, result in an exponential blow-up in formula size. We introduced the notion of a substitution-program, which was intended to give a more compact representation for formulas. We showed that each formula in $\mathrm{FP}(\widetilde{\mathbf{\Omega}})$ can be translated to a P-uniform family of $\mathrm{FO}(\widetilde{\mathbf{Q}}_{\mathbf{\Omega}})$-substitution programs. We used this result to also establish a translation to infinitary logic.

We next introduced a framework for studying symmetric circuits over a much richer class of bases. We noted that the usual definition of a circuit implicitly imposes the assumption that a circuit is defined over a basis of trivially invariant functions. We showed that any P-uniform family of symmetric circuits defined over a basis of symmetric functions must define a query expressible in FPC. In order to go beyond FPC we needed to generalise the circuit model. We developed a framework of structured Boolean functions that take as

input τ-structures rather than strings. We then considered bases of isomorphism-invariant structured functions, a generalisation of the notion of a symmetric function (which in our framework are instead called *trivially invariant functions*). We showed that each family of almost relational generalised operators defines a corresponding basis of structured functions. We generalised the notion of a circuit in order to include for each gate g a structure on the children of g corresponding to the structured function labelling g. We generalised the notion of a circuit automorphism to ensure that this additional structure is preserved, and correspondingly generalised the notion of a symmetric circuit.

While our aim was to generalise many of the important results from [3], we found that in almost every case the techniques they employ fundamentally rely on the assumption that the circuit is defined over a basis of trivially invariant functions. One particular difficulty arose due to the fact that determining if a map preserves for each gate g the structure on the inputs of g requires checking if two structures are isomorphic. As such, we showed that in our general setting, unlike in [3], deciding many crucial circuit properties, such as the syntactic-equivalence relation or the action of an automorphism, is at least as hard as solving the graph-isomorphism problem. This posed a problem for two reasons. First, we needed to be able to transform circuits into a particular normal form in polynomial-time in order to apply the support theorem, and any such translation seems to require these properties be polynomial-time decidable. Second, the translation from circuits to formulas we use explicitly relies on the polynomial-time decidability of a number of related properties, including the orbit of a gate and the action of an automorphism. In response we introduced the notion of a transparent circuit. We showed that on this class of circuits all of the relevant problems are polynomial-time decidable, and so we restricted our main result to transparent circuits.

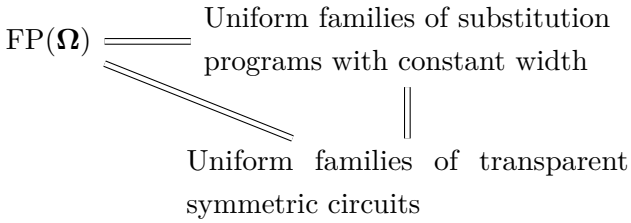
The restriction to transparent circuits makes the translation of uniform circuit families into formulas of the logic (which is the difficult direction of our characterisation) possible, but it complicates the translation in the other direction. Indeed, the natural translation from formulas of fixed-point logic to uniform circuit families yields circuits which are symmetric, but not necessarily transparent. As such, we needed to explicitly define a translation from fixed-point logics to uniform families of circuits. Here we used the translation from $\mathrm{FP}(\widetilde{\mathbf{\Omega}})$ to P-uniform families of $\mathrm{FO}(\widetilde{\mathbf{Q}})$-substitution programs and then showed that by adding a number of gadgets to the circuit we could transform these families of substitution programs into equivalent P-uniform families of circuits.

In order to establish the translation we needed to generalise a number of crucial results proved by [3]. In particular, we generalised the support theorem. The conclusion of [3] says that the support theorem is "largely agnostic to the particular [...] basis", suggesting that it could be easily adapted to include other gates. This turns out to have been a misjudgement. Attempting to prove the support theorem for a basis that includes non-trivially invariant gates showed us the extent to which both the proof of the theorem and, more broadly, the definitions of a circuit, relies on the assumption that all functions computed by gates

are trivially invariant. We developed new techniques that avoided these assumptions and established a generalisation of the support theorem.

We completed our characterisation by showing that P-uniform families of circuits can be translated to formulas in the corresponding extension of fixed-point logic. Here again we noted the extent to which the results in [3] rest on the symmetry assumption on the basis. In order to prove this result we developed a radically new set of tools that allowed us to recursively evaluate the gates of the circuit in the logic without the symmetry assumption, and so established the translation.

In short, we can represent the proof of our characterisation through the three equivalences in this triangle.



This highlights another interesting aspect of our result. We notice that the formulation of fixed-point with rank in [13] can similarly be formalised as an extension of $\mathrm{FP}^{\mathbb{N}}$ by an (infinite) union of vectorised operators $\mathbf{\Omega}$. We note that $\mathbf{Q}_{\mathbf{\Omega}} = \mathbf{Q}_{\mathbf{\Omega}_{\mathbf{rk}}}$ and $\mathbb{B}_{\mathbf{\Omega}} = \mathbb{B}_{\mathbf{\Omega}_{\mathbf{rk}}}$, where $\mathbf{\Omega}_{\mathbf{rk}}$ is the rank operator defined in Section 3.2. However, Grädel and Pakusa showed the fixed-point logic with rank defined in [13] is strictly less expressive than FPR [22]. The fact that we can complete the cycle of equivalences for FPR but not for this less expressive logic suggests that FPR is the "right" formulation of rank logic. We also conclude from this example that the restriction to *finite* unions of vectorised operators in the statement of the main theorem is necessary.

10.2 Future Work

There are many directions for new work suggested by the methods and results of this thesis. First of all, there is the question of transparency. We have established a characterisation of the expressive power of families of transparent symmetric circuits but we are not sure if this model is strictly weaker than without the transparency restriction. If the model is strictly weaker for any basis then the graph-isomorphism problem in not in polynomial-time (see Proposition 9.5). Of course, if we could show that for an appropriate collection of vectorised operators $\mathbf{\Omega}$ the P-uniform families of transparent symmetric circuits over the basis $\mathbb{B}_{\mathbf{\Omega}}$ are strictly less expressive than P-uniform families of arbitrary symmetric circuits over $\mathbb{B}_{\mathbf{\Omega}}$, then we would have separated $\mathrm{FP}^{\mathbb{N}}(\widetilde{\mathbf{\Omega}})$ from P.

We have explored methods of extending the symmetric circuit model by allowing for richer bases. Another approach to extending the model would be to consider relaxations of

the symmetry property. There are numerous ways we might do this. We could consider, for example, circuits with the property that all of the permutations from a subgroup $G \leq \mathbf{Sym}_n$ extend to automorphisms of the circuit. We say such circuits are *semi-symmetric relative to* G. We notice that the circuits that are semi-symmetric relative to the full permutation group are exactly the symmetric circuits and the circuits that are semi-symmetric relative to the trivial group are the circuits with no symmetry restriction at all. In this way by considering subgroups of the symmetric group of increasing index we can interpolate from the extension of fixed-point logic (such as FPC or FPR) to P.

This also offers an interesting approach we might use to develop a circuit characterisation of choiceless polynomial time (CPT) [7]. CPT is one of the most powerful logics known to be contained in polynomial-time but not known to be strictly contained [14, 1]. We attempted to develop a circuit characterisation of CPT within the framework presented in this thesis, but found that CPT seemed to require a different, perhaps weaker, notion of symmetry. We might start by seeing if the formulas in CPT naturally define semi-symmetric circuits relative to some subgroup. Alternatively, we could start with a known CPT algorithm, e.g. from [14], and see if we can define a semi-symmetric circuit from one of these algorithms. The prospect of a circuit characterisation of CPT is certainly an intriguing one. Not only would it clarify the role of symmetry in understanding CPT, but bringing FPR and CPT into the same framework might allow us to begin to study the relationship between these logics, and perhaps separate them.

There are still more ways of considering weakenings of the symmetry condition. In order to define our more general notion of a circuit we needed to relax the usual requirement of symmetry on the inputs of a gate. We could weaken this condition further and instead consider functions of the form $f : \{0,1\}^X \to \{0,1\}$ where X is an arbitrary set, along with an explicit invariance condition given by a subgroup $G \leq \mathbf{Sym}_X$. In order to work with these circuits we would first need to prove a result analogous to the support theorem. In this framework it would be natural to use a more fine-grained approach and treat asymmetry as a resource, which we could then prove lower bounds against. A first step in this direction might be to consider circuits with only a constant or logarithmic number of gates labelled by non-trivially invariant functions.

Both of these approaches involve "relativising" the symmetry notion in some way to a subgroup of the full permutation group. In both cases we believe it can be shown that relativising the symmetry notion to a subgroup of polynomial order results in a circuit model with the same expressive power as the circuit model without any symmetry restriction. Similarly, for subgroups of polynomial size index the relativised symmetry notion results in a model with the same expressive power as that with the full symmetry requirement. As such, the interesting questions would be about "intermediate" subgroups of the permutation group. The structure of these groups is largely unknown, but it would be a natural first step to ask what sorts of structural results would be needed in order to separate circuits with symmetry

properties relativised to different subgroups. This direction, although speculative, may help establish connections between logic, complexity theory, and permutation group theory.

We have thus far discussed three ways of extending the circuit model, the above two approaches for relaxing symmetry and the approach discussed in this thesis of allowing for a richer basis. These three, along with the possibility of varying the uniformity condition on the circuits, provide a rich space of possibilities to explore. It would be particularly interesting to consider how these various approaches, which seem to pull in somewhat different directions, might trade-off against one another.

The circuit characterisations of a logic emphasises certain combinatorial parameters against which we could prove lower bounds. One such parameter is the fan-in of the gates of the circuit. A promising and novel approach would be to try and prove lower bounds for symmetric circuits with gates with bounded fan-in. A concrete first question would be to ask if it is possible to compute AND[3] using a symmetric circuit with gates that have fan-in two. We also do not yet have any general techniques for proving lower bounds for symmetric circuits. It might be useful to consider how techniques used for establishing inexpressibility in logic might be directly used to prove lower bounds for symmetric circuits. Dawar [12] has already shown how the bijection games of Hella [27] can be directly used to prove lower bounds for symmetric circuits without reference to the logic. We might ask if the game characterisations of the expressive power of fixed-point logics given by Holm [29] can similarly be applied directly to the circuit model. It would be especially interesting to see if combining these methods with standard circuit-based techniques, such as the switching lemma and random restrictions, could yield new lower bounds.

References

[1] F. Abu Zaid, E. Grädel, M. Grohe, and W. Pakusa. Choiceless polynomial time on structures with small abelian colour classes. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium*, pages 50–62, 2014.

[2] N. Alon and R. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, Mar 1987.

[3] M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, 60(3):521–551, 2017.

[4] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[5] A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666–1683, 2009.

[6] J. Barwise. *Chapter I: Model-Theoretic Logics: Background and Aims*, volume 8 of *Perspectives in Mathematical Logic*, pages 3–23. Springer-Verlag, New York, 1985.

[7] A. Blass, Y. Gurevich, and S. Shelah. Choiceless polynomial time. Technical report, 1997.

[8] R. Boppana and M. Sipser. The complexity of finite functions. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 757–804. 1990.

[9] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[10] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25(1):99 – 128, 1982.

[11] A. Dawar. Generalized quantifiers and logical reducibilities. *Journal of Logic and Computation*, 5(2):213–226, 1995.

[12] A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.

[13] A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *2009 24th Annual IEEE Symposium on Logic In Computer Science (LICS)*, pages 113–122, 2009.

[14] A. Dawar, D. Richerby, and B. Rossman. Choiceless polynomial time, counting and the Cai-Fürer-Immerman graphs. *Annals of Pure and Applied Logic*, 152(1):31–50, 2008.

[15] A. Dawar and G. Wilsenach. Symmetric circuits for rank logic. `arXiv:1804.02939`.

[16] A. Dawar and G. Wilsenach. Symmetric circuits for rank logic. In *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, pages 20:1–20:16, 2018.

[17] L. Denenberg, Y. Gurevich, and S. Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control*, 70(2):216–240, 1986.

[18] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Kar, editor, *Complexity of Computations*, pages 43–73. AMS, 1974.

[19] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, Dec 1984.

[20] E. Grädel, P. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2007.

[21] E. Grädel and M. Otto. Inductive definability with counting on finite structures. In *Computer Science Logic, 6th Workshop, CSL '92, San Miniato, Italy, September 28 - October 2, 1992, Selected Papers*, pages 231–247, 1992.

[22] E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! In *2015 24th Annual Conference on Computer Science Logic, (CSL)*, pages 390–404, 2015.

[23] M. Grohe. Fixed-point logics on planar graphs. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 6–15, 1998.

[24] M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. In *2010 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 179–188, July 2010.

[25] M. Grohe and J. Mariño. Definability and descriptive complexity on databases of bounded tree-width. In *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings.*, pages 70–82, 1999.

[26] Y. Gurevich. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science*, pages 1 – 57. Computer Science Press, Rockvill, 1988.

[27] L. Hella. Logical hierarchies in PTIME. *Information and Computation*, 129(1):1 – 19, 1996.

[28] L. Hella, P. Kolaitis, and K. Luosto. Almost everywhere equivalence of logics in finite model theory. In *Bulletin of Symbolic Logic*, pages 422–443, 1996.

[29] B. Holm. *Descriptive complexity of linear algebra*. University of Cambridge, 2010.

[30] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1-3):86 – 104, 1986.

[31] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proceedings of the Second Annual Conference on Structure in Complexity Theory, Cornell University, Ithaca, New York, USA, June 16-19, 1987*, 1987.

[32] N. Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer New York, 1999.

[33] S. Jukna. *Boolean Function Complexity: Advances and Frontiers*. Algorithms and Combinatorics. Springer Berlin Heidelberg, 2012.

[34] P. Kolaitis and J. Väänänen. Generalized quantifiers and pebble games on finite structures. *Annals of Pure and Applied Logic*, 74(1):23 – 75, 1995.

[35] P. Kolaitis and M. Vardi. Infinitary logics and 0–1 laws. *Information and Computation*, 98(2):258 – 294, 1992.

[36] L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2004.

[37] P. Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32(3):186–195, 1966.

[38] M. Otto. *Bounded Variable Logics and Counting: A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Cambridge University Press, 1997.

[39] M. Otto. The logic of explicitly presentation-invariant circuits. In *1996 10th International Workshop, Annual Conference on Computer Science Logic (CSL)*, pages 369–384. Springer, Berlin, Heidelberg, 1997.

[40] B. Rossman. On the constant-depth complexity of k-clique. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 721–730, 2008.

[41] J. Rotman. *An Introduction to the Theory of Groups*. Graduate Texts in Mathematics. Springer New York, 1999.

[42] M. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 137–146, New York, NY, USA, 1982. ACM.

[43] H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Springer-Verlag, Berlin, Heidelberg, 1999.

[44] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, May 1985.

Index of Terminology